Treball de Fi de Grau

## Grau en Enginyeria en tecnologies industrials

# Visualització d'escenes de Realitat Virtual amb ulleres HTC Vive amb interacció

**MEMÒRIA**


**Autor:** Laiho Passols, Ari-Pekka
**Director:** Susin Sanchez, Toni

**Convocatòria:** Febrer 2017



Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona


# Resum

El treball està enfocat a la creació d'escenes d'immersió a la realitat virtual mitjançant el motor de videojocs multi plataforma *Unity* i el conjunt de dispositius de realitat virtual *HTC Vive*. En aquest treball es busca avançar amb les tècniques de programació i creació d'escenes amb l'editor de *Unity*.

S'ha plantejat un sistema de treball enfocat a la creació i disseny de tres aplicacions que investiguen diferents camps d'estudi dins del món de *Unity*, la programació amb *HTC Vive* i la interacció de les escenes amb els usuaris. Una de les tres aplicacions s'enfoca en el desenvolupament de noves eines de software pel dispositiu *HTC Vive*, una altra s'enfoca a la realització d'escenes per posar en pràctica els coneixements adquirits sobre *Unity* i, finalment la última es centra en el disseny d'un *serious game* d'ajuda a la rehabilitació de persones amb problemes motrius mitjançant la immersió virtual.

Cada aplicació treballa amb la seva pròpia sistemàtica la idea principal de millora tecnològica i amb les tres es dona sentit a la totalitat del projecte.

Per aconseguir l'objectiu del projecte s'ha endinsat amb la programació utilitzant el llenguatge *C#* i amb l'entorn de treball de l'editor de *Unity*. En la memòria es posa en context l'evolució de la tecnologia dels dispositius de realitat virtual de l'actualitat, s'explica la tecnologia del sistema *HTC Vive*, s'explica com utilitzar l'editor de *Unity* per programar un videojoc i s'explica com treballar i programar l'editor per la creació d'un videojoc de realitat virtual amb *HTC Vive*.

# Sumari



ETSEIB

# 1. Glossari

**Asset:** Paquets de dades. Agrupacions de fitxers de *Unity* que es poden importar a les escenes virtuals com a suport al desenvolupament de les aplicacions.

**Classe (informàtica):** És una plantilla de programes codificats per crear objectes de forma més ràpida i fàcil. Gràcies a les classes amb una sola paraula es poden invocar variables i funcions que d'una altra forma s'haurien de programar a mà.

**C#:** Llenguatge de programació.

**DC:** *"Direct current"*, corrent contínua.

**Dev kit:** *"Development kit"*, kit de desenvolupament. Conjunt d'eines que li permet al programador desenvolupar aplicacions per cada tipus de plataforma.

**Display:** Dispositiu d'un aparell electrònic o pantalla on es visualitza certa informació.

**Dispositiu de *gaming*:** Dispositiu per jugar a videojocs.

**Driver:** Controlador de dispositiu, programa que permet al sistema operatiu d'un ordinador interaccionar amb un perifèric.

**Frame:** Moment de actualització de la imatge en una pantalla de visualització.

**Framerate:** Imatges que actualitza una pantalla al segon.

**Hardware:** Conjunt d'element físics que formen part d'una computadora o sistema informàtic.

***HMD*:** *"Head mounted display"*, ulleres de realitat virtual.

**Input:** Conjunt de dades que s'introdueixen a un programa informàtic.

**Java:** Llenguatge de programació.

**Latència:** La suma de retards temporals dintre d'un sistema electrònic.

**Plug-in:** Aplicació a programa informàtic que es relaciona amb un altre programa per agregar-li una funció nova i generalment molt específica.

**Percepció:** Mecanisme mitjançant el qual els nostres sentits processen informació en forma d'energia que ens arriba des de l'exterior del nostre cos: és a dir: des de l'ambient on ens movem.

**Resolució de pantalla:** Número de píxels que pot mostrar una pantalla.

**Renderitzar:** Representar virtualment una imatge o una escena, generalment en tres dimensions, simulant-ne els efectes òptics de llum, ombra, color, textura o moviment a partir de les dades d'un model computacional.

**RV:** Realitat virtual.

**Set-up:** Passos previs a realitzar per deixar un sistema llest pel seu funcionament.

**Script:** Programa usualment simple se normalment s'emmagatzema en un arxiu de text.

**Software:** Conjunt de programes i rutines que permeten a una computadora realitzar determinades tasques.

**Tracking:** Rastreig. Obtenció de la ubicació i posicionament d'un objecte.

**UI:** *"User interface"*, interfície d'usuari. Medi amb el que l'usuari pot comunicar-se amb una màquina, un equip o una computadora.

***Unity*:** Motor de videojocs multi plataforma creat per *Unity Technologies.*

***VMR:*** *Virtual Motor Rehabilitation*, joc de rehabilitació virtual.

# 2. Prefaci

## 2.1 Origen del projecte

Aquest treball forma part dels quatre que inicien amb la proposta del departament de Realitat Virtual de la Facultat de Matemàtiques i Estadística (*FME)* de la *UPC* de Barcelona, per crear una línia de projectes de llarga durada els quals millorin les experiències d'immersió a la realitat virtual.

El sistema proposat està plantejat pel centre de Realitat Virtual *CRV* que es troba dins de la facultat *FME* i pretén crear una comunitat d'alumnes els quals, a mesura que passin els quadrimestres, avancin cap a un millor desenvolupament i disseny de la tecnologia d'immersió virtual, per a tal fi, està pensat que els estudiants un cop acabada la feina deixin documentats tots els avanços aconseguits a disposició de la Universitat de forma clara i concisa, amb la finalitat de que les persones que comencin un projecte nou es puguin nodrir de la documentació prèvia realitzada. Creant un sistema com el plantejat, es vol facilitar eines als estudiants que inicien els seus projectes amb la finalitat d'afavorir la recerca d'informació, per així trobar noves tècniques i coneixements innovadors per millorar les experiències d'immersió a la realitat virtual any rere any.

## 2.2 Motivació

Un dels motius principals per l'elecció d'aquest projecte ha sigut el moment tecnològic en el que s'ha realitzat el treball. Estem en una societat on cada vegada s'avança més de pressa en totes les vessants de la tecnologia electrònica, ja sigui en hardware o software. Ens trobem en un punt d'expansió i innovació tecnològica on els dispositius es renoven en períodes molt curts de temps i, si es vol estar al dia dels sistemes més punters, s'ha de seguir la corrent de desenvolupament de les empreses de molt a prop per no quedar desfasat. Dins de la programació i la creació d'aplicacions, és un privilegi poder treballar amb la realitat virtual just quan comença a consolidar-se al mercat i a ser reconeguda per la societat. La tecnologia ha evolucionat fins el punt de poder aportar les necessitats i especificacions tècniques que el sector de la realitat virtual necessitava i ja es poden crear experiències que cobreixen les expectatives de l'usuari.

A part, el paquet de dades *SteamVR* de *Unity* que s'utilitza per programar pel conjunt de dispositius de *HTC Vive* està en plena fase de desenvolupament i no hi ha millor moment per descobrir com utilitzar l'eina i el seu funcionament intern, doncs poques vegades es pot estudiar de tant a prop els inicis d'un software d'una gran companyia com es *Valve* i és molt gratificant poder avançar conjuntament amb els desenvolupadors de *SteamVR* cap a una nova forma de crear aplicacions a *Unity*.

## 2.3 Requeriments previs per realitzar el projecte

En un treball d'aquestes característiques es té molta llibertat respecte la metodologia d'aprenentatge i la búsqueda de fonts d'informació. Un dels avantatges de treballar amb *Unity* és que hi ha una gran comunitat d'usuaris al darrere i per aquest motiu si algú es vol endinsar en un projecte semblant, no són necessaris molts coneixements previs per poder-lo realitzar: tot es pot anar aprenent a mesura que s'avança en el desenvolupament del treball si es tenen ganes i s'és constant. No obstant, sí que és veritat que s'han de tenir uns mínims coneixements de programació per poder realitzar els scripts que es necessitin durant la creació de les aplicacions. Hi ha molta documentació per la xarxa i és fàcil ser autodidacta si un mateix es prepara i li dedica temps a la búsqueda d'informació. A part, la mateixa empresa de *Unity* s'ha dedicat a penjar a la xarxa obertament tutorials i informació sobre el programa perquè sigui més fàcil pels principiants començar a endinsar-se en l'entorn de treball.

## 2.4 Enfocament donat al treball

El treball està enfocat en l'obtenció d'informació i coneixements entre el programa *Unity* i el conjunt de dispositius *HTC Vive* a través de la creació d'aplicacions d'immersió virtual. En el cas d'aquest treball s'han volgut repartir les tasques amb la creació d'un conjunt de tres aplicacions en les quals en cada una d'elles es treballen i milloren diferents camps d'estudi. Cada aplicació treballa amb la seva pròpia sistemàtica l'idea principal i amb el conjunt de les tres es forma la totalitat del projecte. Al ser un projecte amb moltes vessants a estudiar i aplicar, implementar una diversificació de les tasques a realitzar és la millor forma d'enfocament per poder obtenir una estructura de projecte compacta.

S'han plantejat des del començament tres perspectives ben diferenciades:

- La primera la de l'estudi i pràctica del software *Unity* que s'utilitza en el projecte.
- La segona la de la creació de d'eines bàsiques per interactuar amb el sistema de *HTC Vive* que serveixin com a suport pels estudiants que s'iniciïn amb el plug-in *SteamVR* a *Unity.*
- La tercera la de l'aplicació dels coneixements adquirits per crear un *serious game* que es centri en l'ajuda a la rehabilitació de les persones amb problemes de capacitat motores.

Aquestes aplicacions han desencadenat en un joc de *Unity* anomenat *Balls Game* que ha servit per:

- Millorar les tècniques de creació d'escenes virtuals.
- Una aplicació on hi han scripts executables amb *HTC Vive* anomenada *HTCViveStudentsCRV.*
- El *serious game* de rehabilitació amb realitat virtual anomenat *MotrizVR*.

* Més endavant s'explica el funcionament, la creació i el disseny de les tres.

# 3. Introducció

## 3.1 Objectius del projecte

L'objectiu principal d'aquest treball és el d'obtenir informació i coneixements entre el programa *Unity* i el conjunt de dispositius *HTC Vive,* per iniciar una línia de projectes universitaris sobre la immersió a la realitat virtual de llarga durada. Per arribar a l'objectiu principal s'han d'aconseguir crear vàries escenes d'interacció amb les ulleres de realitat virtual. Per poder assolir la tasca principal apareixen altres objectius secundaris que són essencials per poder completar la feina.

Primer de tot i pensant amb els projectes vinents, s'ha de realitzar un informe complet sobre les tècniques emprades a l'hora de crear les escenes i els scripts del treball, d'aquesta manera al tenir una documentació ordenada de tots els apartats del projecte, més endavant es podrà utilitzar aquest document com a referència a seguir pels nous estudiants.

## 3.2 Abast del projecte

En aquest projecte s'ha realitzat una aplicació gradual de les tècniques de desenvolupament de software amb el motor *Unity* a mesura que s'obtenien més coneixement. Primer de tot s'ha realitzat un videojoc complet, amb tècniques bàsiques però de molta diversitat. Després, s'ha adaptat el videojoc en un entorn de realitat virtual per a les ulleres *HTC Vive*.

Al tenir els coneixement bàsics sobre la implementació de *HTC Vive* a l'editor de *Unity,* s'han creat eines bàsiques d'interacció pels dispositius del sistema de realitat virtual. Per finalitzar s'ha creat un *serious game* que enllaça tots els coneixement obtinguts durant del treball, des de les tècniques apreses en un començament fins a les últimes eines de suport creades pels dispositius.

Totes aquestes tasques, s'han realitzat amb una documentació clara i concisa perquè serveixi de referència i suport als alumnes.

# 4. Context històric de la tecnologia de RV

La idea de crear unes ulleres de realitat virtual no és nova, ja fa temps que s'anhela un invent com aquest però no s'havia pogut dur a terme abans degut a la capacitat tecnològica que es necessita. No només s'han de crear els efectes de percepció tridimensional a partir d'imatges que han de semblar reals i han de ser renderitzades al moment, també s'han de poder enregistrar i sincronitzar la posició i el moviment dels dispositius d'una forma precisa i fluïda per donar un efecte creïble. Les ulleres de realitat virtual que existeixen avui en dia (ja siguin les *HTC Vive* utilitzades en aquest projecte o altres semblants) han desencadenat en el que són degut a una evolució d'anys de descobriments i millores tecnològiques iniciades abans del que molts es poden imaginar. Es podria dir que la noció de l'estereoscòpia i el naixement dels computadors digitals, juntament amb la seva ràpida i continua millora, són els dos pilars en els que es sosté la tecnologia de realitat virtual tal i com es coneix avui en dia.

## 4.1 La visió humana i la tècnica de l'estereoscòpia

La humanitat sempre ha tingut interès en aprendre com funciona la morfologia del cos humà i entre aquests coneixements estava la qüestió de saber com es processava la informació a través del sistema visual. El 80% de la informació que es rep de l'entorn arriba a través de la visió, la percepció visual és, per tant, el procés que recull més dades de l'entorn amb el que s'interactua. Durant l'època renaixentista, artistes i matemàtics italians del segle van emprendre la investigació en el camp de la percepció visual per poder recrear millor les seves obres d'art. Amb aquesta voluntat van començar a investigar el fenomen de la perspectiva mitjançant solucions geomètriques i es van endinsar en la recreació d'escenes tridimensionals en superfícies planes. Filippo Bruelleschi (1377-1446) de Florència va ser un dels primers en manifestar l'art de la perspectiva linear per crear les seves obres d'art, però encara sense utilitzar principis geomètrics. A l'any 1435 Leon Battista Alberti (1404-1472) va publicar la primera obra on es parlava sobre la tècnica i utilització de la perspectiva geomètrica anomenada *Della Pittura* i més endavant altres renaixentistes com Leonardo da Vinci (1452-1519) van millorar els mètodes de treball. Precisament va ser da Vinci qui es va adonar per primera vegada d'un dels fenòmens crucials pel funcionament dels sistemes de realitat virtual, el de la visió binocular, un fenomen que permet als essers humans captar la

profunditat de l'entorna través de la vista. Els pensaments de da Vinci sobre el tema van quedar reflectits en el llibre *Trattato della Pittura* publicat per primera vegada l'any 1651 [2].

*"Un dibuix, encara que fos recreat amb el millor art i finalitzat amb absoluta perfecció, a més dels contorns, les llums, les ombres i els colors, mai podria representar un relleu igual al dels objectes reals llevat que aquest es mirés des de distància i amb un únic ull." [Leonardo da Vinci, Trattato della Pittura 1651].*

### 4.1.1 La visió binocular

Estrictament parlant tots els animals amb dos ulls tenen visió binocular, al tenir dos òrgans receptors d'imatges diferents tots ells tenen camps visuals independents a cada ull que després el cervell fusiona. A l'unir-se la informació rebuda de l'exterior hi han regions de l'entorn que es veuen per duplicat i com a conseqüència es solapa una de les zones de les imatges que es processen. No obstant, no tots els animals amb dos ulls perceben l'entorn de la mateixa manera, el terme de visió binocular sol referir-se específicament als animals amb la peculiaritat de poder captar la profunditat de l'entorn, aquests animals són els que tenen una gran regió on es superposen els camps visuals de les dues imatges que enregistren els ulls, regió que aprofita el cervell per codificar la profunditat.

A la *Figura 4.1* es representen les fronteres del camp de visió dels humans. Cada ull té el camp limitat a una certa zona (l'esquerra des de A fins a C, el dret des de D fins a B) i a la regió on es superposen els dos camps de visió (el comprés entre B i C) és té la zona de visió binocular que processa el cervell per crear la percepció de profunditat. La visió binocular dels essers humans es pot extrapolar en matisos molt generals al mateix sistema de percepció espacial que es processa amb el sentit de l'oïda. Els dos reben informació des de dos canals diferents que després el cervell sintetitza per treure una sola resposta; en el cas de la visió són els ulls (la retina) i en el cas l'oïda ho són les orelles (timpà).

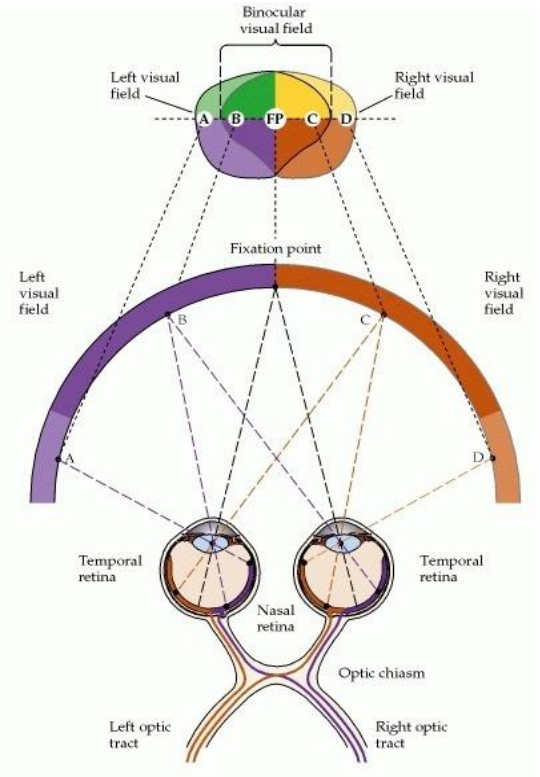



*Figura 4.1: Camp de visió del sistema visual humà*

Els dos sentits tenen una fase del sistema de posicionament espacial molt semblant, amb l'oïda es poden localitzar des de quina zona provenen els sons que s'escolten, es pot saber si són llunyans o propers i es pot saber des de quina direcció s'emeten, d'aquesta manera es posiciona la senyal emesa de forma aproximada a l'entorn, el mateix sistema utilitza la vista per percebre la profunditat de l'espai que es visualitza. La comparació dels dos canals d'informació és essencial per poder ajustar correctament els objectes a l'espai.

Després de la investigació de Leonardo da Vinci sobre la visió binocular es va continuar avançant amb l'estudi de la visió tridimensional però no va ser fins l'any 1838 quan va tornar a succeir un pas molt important en la història de la tecnologia d'immersió virtual, Sir Charles Wheatstone (1802-1875) de Gloucester, famós científic i inventor del Regne Unit, va definir els principis d'una nova vessant de la tecnologia que seria la base de tot sistema de visualització tridimensional, la tècnica de l'estereoscòpia. Va inventar el primer aparell aplicant aquesta nova tècnica que va anomenar estereoscòpic, un aparell que va ser crucial per inaugurar la investigació moderna de la visió estereoscòpica i que seria la base de les ulleres de realitat virtual.

### 4.1.2 L'estereoscòpia

El terme de visió estereoscòpica es refereix al conjunt processos que segueix el cervell per recrear estructures tridimensionals a través de la vista, la paraula prové del grec *"stereopsis"* (στερεο -stereo- "sòlid" i ὄψις-opsis- "aparença") que es podria traduir literalment com a aparença sòlida de la visió. Es sol utilitzar la paraula principalment per parlar sobre la percepció de profunditat i solidesa dels objectes en la visió binocular. Wheatstone va anomenar als seus aparells estereoscòpics en referència a la paraula "stéréoscopique" que havia utilitzat François Aguilonius (1567-1617) per denotar la visió binocular.

Un dels primers estereoscòpics de la història es mostra a la *Figura 4.3*, consistia de dos miralls col·locats i girats amb uns angles concrets i dues bases per aguantar les fotografies o dibuixos que es volien utilitzar. En el seu primer prototip Wheatstone va utilitzar parells d'imatges de figures geomètriques *Figura 4.4* que ell mateix havia dissenyat. Tot aparell estereoscòpic té la funció de transmetre dues imatges prèviament preparades per l'efecte, si s'aconsegueix que amb el dispositiu cada ull visualitzi només una de les dues imatges sense cap mena d'interferència és quan el cervell processa la percepció estereoscòpica.

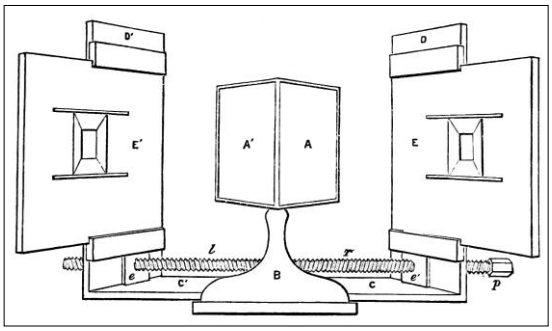



*Figura 4.3: Primer estereoscòpic creat per Wheatstone.*

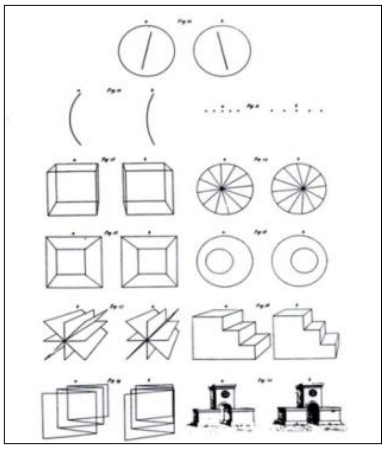


Figura 4.2: *Imatges empleades per l'estereoscòpic.*

Mitjançant els miralls de l'estereoscòpic es podien controlar els angles de convergència de visualització dels ulls i, gràcies a això, es podien realitzar experiments amb variacions dels inputs i s'era capaç de trobar les correlacions entre els angles de convergència donats i la percepció de la profunditat. La investigació de Wheatstone amb l'estereoscòpic el va fer arribar a la conclusió de que la sensació de profunditat no només depenia de la disparitat de les imatges sinó que també depenia dels angles de convergència.

Wheatstone va reportar els detalls de la construcció del seu invent l'any 1838 a la societat científica de Londres *Royal Society* i va fer una demostració on estaven presents molts científics de l'època, entre ells Sir David Brewster (1781-1868) secretari de la *Royal Society,* que després de finalitzar la presentació va comprar un model de l'aparell per començar els seus propis experiments. Brewster va crear la seva pròpia versió de l'estereoscòpic i va millorar la forma de visualització del sistema, el seu invent consistia en una caixa on es visualitzaven les dues imatges a través de prismes, aquests prismes eren lents convexes que realitzaven la mateixa tasca de desviació visual que els miralls d'una forma més còmode per l'usuari, *Figura 4.4*.

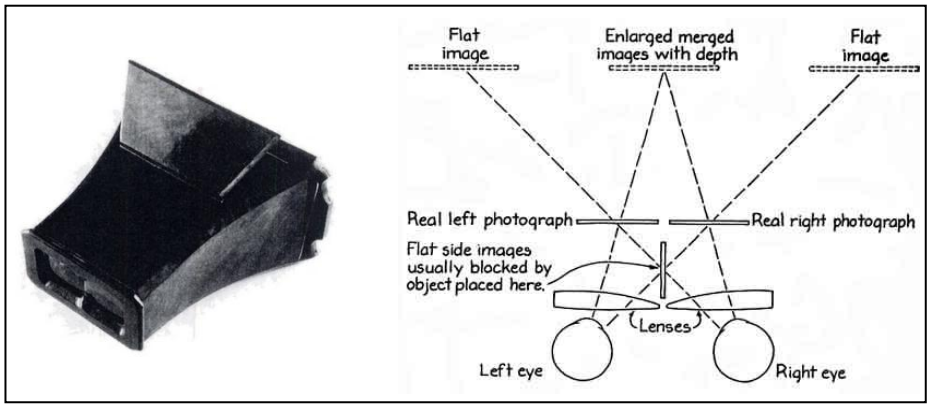



*Figura 4.4: Imatge i esquema de funcionament de l'estereoscopi creat per Brewster.*

## 4.2 Els avanços de la realitat virtual als inicis de l'era digital

Un segle més tard de la construcció dels primers estereoscòpics, a la dècada del 1950 fins a finals del 1970 va succeir un canvi històric que va afectar a tota la societat de l'època, la revolució digital o també anomenada tercera revolució industrial. Va aparèixer el nou sector de la tecnologia digital gràcies a l'arribada i la proliferació dels computadors digitals i es van crear molts invents innovadors que revolucionarien el mercat i la vida tal i com es coneixia.

Els primers antecedents de la realitat virtual de l'època provenen del cinema. La irrupció de la televisió a la societat com a gran medi de comunicació de masses, va causar que els grans estudis cinematogràfics plategessin noves formes d'atracció al públic i van iniciar la creació d'espectacles cinematogràfics a gran escala. La línea d'enfocament que van seguir els cineastes es va anomenar "experiència directa", l'experiència directa consistia en fer creure als espectadors que es trobaven dins de la pel·lícula aplicant qualsevol tècnica possible per arribar a aquesta finalitat, així va ser com va arribar el començament de la comercialització dels primers sistemes estereoscòpics, un començament que va estar marcat per les primeres pel·lícules estereoscòpiques projectades als cinemes. Malauradament el sistema tecnològic que s'utilitzava per visualitzar les pel·lícules obligava als espectadors a portar unes ulleres polaritzades de vidre que va portar a dos problemes generalitzats, el primer va ser que no eren prou còmodes i el públic no les acabava d'acceptar i el segon que les ulleres tenien un elevat cost, com a conseqüència d'aquests problemes les pel·lícules estereoscòpiques es van deixar d'emetre uns anys més tard.

Mentre passaven aquests esdeveniments en el món del cinema, en el sector de l'entreteniment va arribar a l'any 1962 el dispositiu que donaria els primers passos a la creació de dispositius individuals de realitat virtual enfocats als usuaris, l'inventor i cineasta Morton Leonard Heilig (1926-1997) va crear la primera màquina de realitat virtual de la història anomenada *Sensorama (Figura 4.5)*.


*Figura 4.5: Propaganda del dispositiu de realitat virtual Sensorama.*

La màquina era un dispositiu que Heilig descrivia com un entorn d'experiència multi sensorial i va suposar un dels primers simuladors d'entorns que existien. Era un intent d'immersió completa en un entorn de realitat virtual amb altaveus estèreo, visualització estereoscòpica per representar imatges tridimensionals, ventiladors per crear corrents d'aire, generadors d'aromes mitjançant bosses que desprenien olors sincronitzadament amb les imatges i una cadira vibratòria. La idea de Heilig malgrat ubicar-la en una època de necessitats de nous invents en el camp de l'entreteniment no va ser acollida pels gran inversors i malauradament Heilig després de produir i editar sis pel·lícules per la màquina, no es va aconseguir suficient ajuda financera i el projecte es va parar abans de comercialitzar-lo. A part de la màquina *Sensorama* Heilig també va patentar i construir un prototip de la màquina però en versió portàtil anomenat *Mascara telefèrica (Figura 4.6)*, el prototip que va ser el predecessor de les primeres ulleres virtuals o *HMD* (*Head mounted displays*) utilitzades actualment, unes ulleres de visió estereoscòpica que serien un avanç del que moltes empreses pioneres en el sector de la immersió virtual realitzarien més endavant.

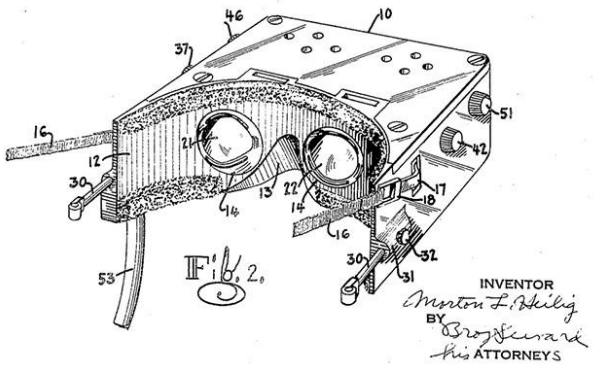



*Figura 4.6: Patent de la "Màscara telefèrica" de Heilig.*

El projecte de Heilig de construcció d'un medi d'experiències multi sensorials no va semblar prou gratificant i còmode pels usuaris. No obstant, a mesura que s'avançava amb la tecnologia digital i computacional, la idea de tenir un món virtual d'interacció amb l'usuari es veia cada cop més viable, aviat es va fer evident que es necessitava una interfície que utilitzés la percepció humana per facilitar la comunicació entre home-màquina si es que es volia prosperaren les investigacions i desenvolupaments.

### 4.2.1 L'arribada dels *HMD* (Head mounted displays)

Al 1968, Ivan Sutherland, programador, professor, informàtic i pioner d'Internet, quan estava treballant al *MIT* (Massachusetts Insitute of Technology) va crear les primeres ulleres de realitat virtual o *Head mounted displays* tal i com les coneixem avui en dia. L'aparell s'anomenava "*Espasa de Damocles"* i mostrava imatges estereoscòpiques a partir de la informació que es processava amb un programa computacional.

Sutherland abans de dissenyar les ulleres havia realitzat una tesis doctoral titulada "*Sketchpad: A man-machine graphical communication system*" al MIT l'any 1963 en la que va crear un sistema anomenat *Sketchpad,* un sistema que habilitava a les persones a utilitzar valors de bits per controlar l'aparença dels píxels que es mostraven a la pantalla d'un ordinador. L'objectiu de la seva tesis era fer mons virtuals que es poguessin construir a la memòria d'un ordinador i com ell es referia *"Semblessin reals, servissin com un mirall al país de les meravelles matemàtiques i tinguessin tots els cinc sentits involucrats" [2]*.

Així que Sutherland va dissenyar un software que culminés el seu objectiu d'un món virtual i utilitzant l'ordinador *TX-2*, aparell basat en transistors, pare dels ordinadors actuals, va crear una sèrie de programes 2D interactius que consistien en dibuixar línies i modificar-les a temps real amb un llapis tàctil *Figura 4.7*. El *Sketchpad* donava la possibilitat als operadors de crear models visuals complexos sobre una pantalla només movent les mans i va ser una innovació que va influenciar a pensar en altres formes alternatives d'interacció amb les màquines que no fos només amb teclat i botons. El camp del disseny assistit per ordinador conegut actualment com a CAD va sorgir d'aquesta tesis i s'ha convertit en un dels motors més importants pel desenvolupament i visualització de models matemàtics bidimensionals i tridimensionals dels últims 90 anys, també ha sigut la base per crear les geometries i models de la realitat virtual més endavant.

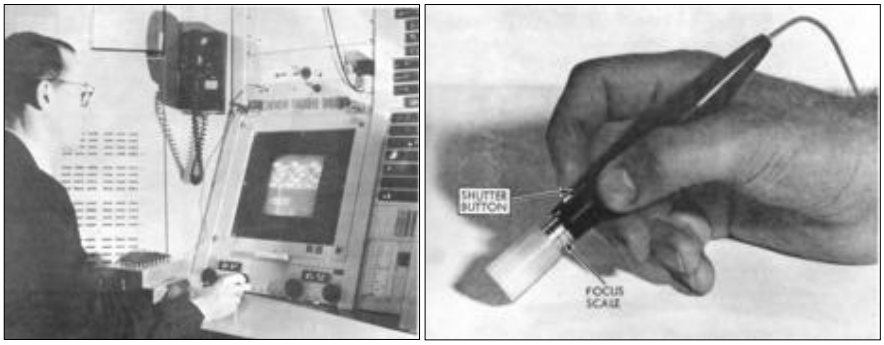



*Figura 4.7 Fotografies de l'aparell Sketchpad i el llapis tàctil que s'utilitzava.*

Sutherland va començar a treballar amb la percepció humana, la tecnologia dels sensors de posicionament, la generació d'imatges a temps real i la total integració del conjunt de paràmetres humans dins del sistema. Va deixar clar a la seva tesis doctoral que l'ésser humà havia de tenir una interacció el més real possible amb l'espai virtual computacional. Per poder fer un sistema d'immersió virtual òptim en la seva opinió el sistema havia de ser multi sensorial, havia de tenir tots els sensors amb tots els graus de llibertat possibles, havia d'ensenyar imatges nítides i complexes amb una baixa latència i a més, la integració de qualsevol paràmetre dins d'aquest entorn no havia de suposar cap tipus de retard al transmetre les dades (no hi podia haver latència).

Amb les seves idees enfocades, Sutherland, després de realitzar la seva tesis doctoral va encaminar el futur de la realitat virtual amb el primer *HMD* computacional de la història, l'*Espasa de Damocles* (mencionada anteriorment). L'*Espasa de Damocles* eren les primeres ulleres de realitat virtual que estaven connectades a un ordinador i a diferència dels altres dispositius podien transmetre la informació processada a temps real a partir de programes informàtics, això donava molta més llibertat de creació d'escenes i donava la possibilitat a realitzar modelats tridimensionals amb variacions de la perspectiva. L'aparell amb l'ajuda de l'ordinador creava una dependència entre el canvi de perspectiva de l'entorn virtual i el posicionament de l'usuari, una idea que s'utilitza ara amb totes les ulleres de realitat virtual que dona una sensació molt més realista d'immersió a l'entorn.

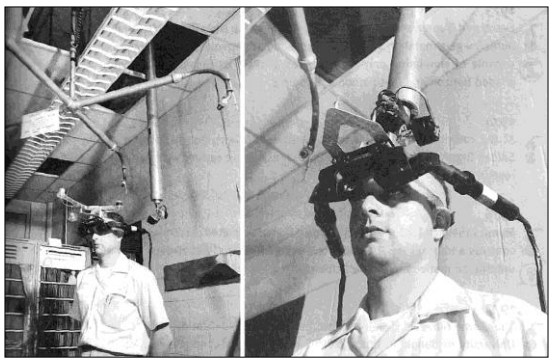


*Figura 4.8: Fotografia de l'Espasa de Damocles.*

L'*Espasa de Damocles (Figura 4.8)* permetia contemplar l'estructura d'un objecte flotant dins de l'habitació podent arribar a veure les diferents cares de l'objecte a mesura que s'avançava per l'espai d'interacció.

Amb aquest prototip va néixer la realitat virtual, el sistema tenia totes les característiques necessàries que s'exigien: estereoscòpia, localització de la posició de l'usuari, possibilitat de navegar al voltant d'un objecte i poder-lo contemplar des de qualsevol zona, immersió de l'usuari a l'entorn d'interacció, etc. Els principals problemes que tenia eren el que més endavant també apareixerien en altres dispositius *HMD* que no van triomfar degut a la limitació tecnològica de l'època, els quals eren la qualitat dels gràfics visualitzats, la resolució de les pantalles de visualització i la fluïdesa de les imatges. Tot i les seves limitacions el concepte que aportava l'*Espasa de Damocles* va ser una de les grans revolucions tecnològiques del sector.

Fins al 1980 la industria de la realitat virtual es va enfocar a donar suport amb els seus aparells al sector dels serveis militars i va quedar estancada en el desenvolupament al mercat dels consumidors de l'època. No va ser fins els anys vuitanta que gràcies a l'abaratiment de les eines de processos i circuits integrats pel disseny d'imatges es va generar un gran impuls en la investigació tecnològica virtual, tan en la síntesis dels gràfics en tres dimensions com en els perifèrics de la realitat virtual. En aquella època també es van descobrir els fractals de Benoit Mandelbrot i el *ray tracing* gràcies a l'empresa MAGI, que consisteix en una tècnica de càlcul pel renderitzat de les reflexions de la llum de les imatges de tota l'escena virtual que es treballa, pixel a pixel. La realitat virtual va ser llavors quan va atraure la cobertura periodística i va sortir dels centres universitaris i militars per ser investigada també per la indústria. Al 1986 Scott Fisher, un investigador que havia estudiat al MIT i més endavant havia treballat fent investigacions sobre les pantalles estereoscòpiques a la companyia Atari, es va incorporar al *Ames Research Center* a la *NASA* per treballar com a director del projecte *VIEW* (Virtual Environment Workstation Project) que tenia l'objectiu de crear un simulador per posar en pràctica l'ajuda al manteniment d'una estació aeroespacial sense haver-hi d'estar present. El projecte *VIEW* es va convertir en el desenvolupament d'un sistema que proporcionava un entorn visual d'imatges estereoscòpiques sensibles a l'entrada de gestos, veu i posició de l'usuari. Permetia a l'operador explorar virtualment un entorn de 360 graus i interactuar amb tots es seus components. Aquest dispositiu (*Figura 4.9)* constava d'una unitat de representació visual estereoscòpica, uns guants per l'entrada de dades amb llibertat de moviments, tecnologia de reconeixement i síntesis de veu, sensors de moviment pels gestos, so tridimensional i un equip per la generació de gràfics per ordinadors i imatges de vídeo.


*Figura 4.9: Imatge del dispositiu creat pel projecte VIEW.*

Al llarg d'aquest projecte van sortir conceptes que formen part, actualment, dels paradigmes de control de les experiències immersives actuals, com per exemple l'escalat del món virtual i la manipulació d'objectes entre altre millores. Posteriorment, l'equip del projecte va construir una càmera de vídeo capaç de proporcionar imatges estèreo a temps real. Amb la càmera i la construcció de l'aparell de *VIEW,* la NASA va investigar la manera d'arreglar els danys d'una estació orbital sense l'obligació d'enviar un alt número d'astronautes, volien recrear la mateixa situació que hi havia a l'estació espacial a l'aparell de realitat virtual *VIEW* a través de la càmera.

Scott Fisher, el director del projecte, tot i estar treballant en aquell moment per la NASA va ser el primer en afirmar que la verdadera transcendència que suposarien les aplicacions de realitat virtual que s'estaven realitzant no serien pel camp aeroespacial, sinó per l'àmbit de la medicina en els simuladors quirúrgics, l'educació i altres àmbits socials. El projecte *VIEW* finalment no es va tirar endavant però els dispositius i les millores tecnològiques aconseguides van ser de molta ajuda per les investigacions que es van fer posteriorment.

### 4.2.2 Els anys noranta, la realitat virtual arriba al mercat del consumidors

A partir dels anys noranta les empreses van veure que hi podia haver un mercat per la realitat virtual en el sector de l'entreteniment i van començar a invertir i desenvolupar dispositius de *gaming* per la societat. Al 1990 el Dr. Jonathan Waldern, qui havia estat investigant la realitat virtual des de el 1985 a la universitat de Loughborough, amb l'ajuda dels laboratoris de IBM va presentar la màquina *Virtuality* (*Figura 4.10*), una màquina recreativa que tenia la principal característica d'unes ulleres de realitat virtual estereoscòpiques per jugar als seus videojocs. Al 1993 *Sega* anunciava el *Sega VR Headset* (*Figura 4.11)*, ulleres de realitat virtual amb pantalles de *LCD*, àudio estèreo i sistema de posicionament, però malauradament no es van poder comercialitzar per dificultats tècniques de desenvolupament.

*Figura 4.10: Imatge de Virtuality.*

*Figura 4.12: Imatge de Sega VR Headset.*

*Figura 4.11: Imatge de Virtual Boy.*

Nintendo també va apostar per la realitat virtual i va crear al 1995 la *Nintendo Virtual boy (Figura 4.11)*, el dispositiu tenia dificultats de visualització ja que només es podien veure les imatges en un sol color i finalment va patir el mateix final que l'aparell de Sega, no va prosperar la idea entre consumidors i l'any següent es deixar la producció i comercialització.

L'etapa dels anys noranta va estar marcada per l'oferta de la realitat virtual per part de les companyies als ciutadans. Els sistemes encara eren massa costosos per l'experiència que oferien i el problema va arribar quan la societat va posar masses expectatives en les experiències que es podrien percebre amb els dispositius, aquestes expectatives no es van complir i van ser contraproduents cosa que va provocar que la gent perdés el interès amb els dispositius de realitat virtual. Encara no hi havia suficient resolució de pantalla, no es podien crear suficients gràfiques, la latència de processament no era l'òptima i això suposava que no fos una immersió completa, encara faltava un pas més d'evolució i optimització tecnològica per arribar a crear un mercat d'ulleres de realitat virtual que prosperés.

### 4.2.3 Avanços tecnològics dels *HMD* al segle XXI

En els últims anys del segle XXI és quan finalment s'ha arribat a l'equilibri entre la tecnologia d'ulleres de realitat virtual que es pot oferir al mercat i el nivell d'immersió que vol el consumidor, els aparells encara tenen un preu elevat perquè tothom pugui tenir un *HMD* a casa seva, però des d'inicis del segle fins a l'actualitat s'ha realitzat un avanç tecnològic impressionant que avui en dia està donant els seus fruits. L'any 2016 ha estat marcat pel llançament de les tres marques més conegudes de *HMD* al mercat, les *Oculus Rift*, les *PlaystationVR* i les *HTC Vive* que s'utilitzen en aquest treball. L'evolució que ha portat a la creació d'aquests dispositius passa principalment per dues estructures bàsiques, la primera la millora de la rapidesa dels processadors per poder dissenyar entorns més realistes i fluids i la segona la millora de la resolució de les pantalles que s'utilitzen com a *displays*. Als dos factors s'hi uneix l'abaratiment que reben cada any els components electrònics dels dispositius, un abaratiment de peces que es trasllada a un producte més competitiu dins del mercat dels consumidors. El inici de les targetes gràfiques actuals, les *GPU* (Graphics Processor Unit) es va iniciar amb la *GeForce256* de *NVIDIA* l'any 1999, una unitat capaç de processar 15 milions de polígons per segon, només al 2006 ja es tenia la *GeForce8800GTX* que podia suportar 36.8 bilions de textures per segon i avui en dia ja es tenen targetes gràfiques que sobrepassen amb molta diferència aquestes capacitats, unes targetes que a més es poden utilitzar per pantalles de millor resolució i que tenen innovadors sistemes tecnològics per millorar el renderitzat de les imatges. Fins aquest últim any la llei de Moore formulada pel cofundador de *Intel* Gordon E. Moore (la qual exposa que cada dos anys es dupliquen el número de transistors en un processador) s'havia complert, en aquest temps s'ha augmentat la capacitat dels processadors d'ordinadors d'una forma increïble. Per posar un exemple numèric,dels 42 milions de transistors que tenia un *Intel Pentium 4 Willamette* l'any 2000 s'ha passat als 2600 milions de transistors d'un *Intel Core i7 Haswell-E (8-core)* l'any 2014. El mateix passa amb la resolució de les pantalles, els primers displays de pantalles *LCD* que portaven les ulleres virtuals eren molt pesats (uns 2kg) i de baixa resolució (360 x 240 píxels), avui en dia es presenten pantalles OLED i AMOLED de 2160 x 1200 píxels (1080 x 1200 cada ull) molt més lleugeres i amb una freqüència de frames per segon o *framerate* de 90fps que dona una fluïdesa d'imatges molt més bona. Encara queda molt camí per avançar, però, amb resolucions de 1080 x 1200 cada ull, capacitats de processament GPU on es poden crear entorn virtuals realistes, baixa latència i bon

*framerate* de processament, ara per ara les escenes artificials que es poden crear donen una sensació d'immersió virtual més que satisfactòria.

Les primeres ulleres modernes *HMD* del mercat es van dissenyar a mans de Palmer Luckey, un jove apassionat de l'electrònica i la realitat virtual i John Carmack, un famós programador de videojocs com *Wolfestein 3D*, *Doom* o *Quake* amb molta experiència en el sector i cofundador de la companyia *id Software*. Luckey ja havia construït les seves pròpies ulleres de realitat virtuals quan una dia va coincidir amb Carmack en un fòrum d'Internet qui, al veure les ulleres li va agradar la idea i va donar suport al projecte, així va ser com més endavant entre els dos van formar l'empresa *OculusVR*. La companyia va aparèixer amb l'ambició de ressorgir la tecnologia de la realitat virtual que la dècada anterior no havia triomfat en el mercat, amb aquest propòsit l'any 2012 van llançar el projecte de construcció d'unes ulleres virtuals que van anomenar *Oculus Rift* a la plataforma web de finançament de projectes *Kickstarter*, una plataforma que ajuda als emprenedors a recaptar fons per poder aconseguir que les seves idees de negoci es puguin fer realitat. Llançar el projecte a *Kickstarter* va ser un èxit que va batre records, en 24 hores ja havien recol·lectat 670.000$ de 2750 persones i en tres dies havien sobrepassat el milió de dòlars de finançament. A partir de l'èxit a Kickstarter van crear un *Dev kit* pels desenvolupadors de software i més endavant, a l'abril del 2014, Facebook va comprar l'empresa per un total de 2000 milions d'euros. Dos anys més tard *Oculus Rift* ja es convertia en una realitat i va es va iniciar la comercialització del producte pels consumidors el mes de març del 2016.


*Figura 4.13: Ulleres Oculus Rift*

Mentre es llançava *Oculus Rift* a *Kickstarter* la companyia de creació de software d'entreteniment *Valve Corporation* també havia començat a treballar en el seu propi disseny de *HMD*, *Valve* va anar avançant amb el desenvolupament del seu producte fins l'etapa on es van veure parats per la limitació tecnològica, tenien un problema amb la qualitat de visualització de les pantalles utilitzades que no podien resoldre sols.

En aquest punt *Valve* es va adonar que necessitava ajuda i van arribar a un acord amb la multinacional productora d'smartphones i tablets *HTC* per arribar a una solució conjunta, del projecte entre *Valve* i *HTC* va sortir el producte *HTC Vive* que van poder llançar al mercat el mes d'abril del 2016.

# 5. Estudi del conjunt dels dispositius de *HTC Vive*

## 5.1 Introducció a *HTC Vive*

L'empresa *HTC* és una marca internacional de productes que va ser fundada l'any 1997, des de els seus inicis l'entitat ha sigut reconeguda per les seves tablets, smartphones i aparells electrònics. Un dels últims aparells afegits a la seva línia de productes ha arribat gràcies a un treball conjunt entre ells i l'empresa *Valve* en forma de dispositiu d'immersió a la realitat virtual. El nou producte s'anomena *HTC Vive* (*Figura 5.1)* i funciona amb un tipus de tecnologia d'immersió virtual anomenada *room-scale,* que habilita a l'usuari a interactuar amb una zona de joc tridimensional on es pot moure amb total llibertat (a diferència d'altres dispositius on només es pot jugar de peu o assegut). El dispositiu està compost d'un casc *HMD* anomenat *Vive Headset* per visualitzar les escenes, dos controladors anomenats *Controllers* per interactuar amb el món virtual i uns components de *tracking* anomenats *Base stations* o *Lighthouses* per poder seguir el posicionament del jugador en tot moment.

*Figura 5.1: Imatge del dispositiu HTC Vive*

Les principals característiques a destacar del dispositiu a nivell d'especificacions tecnològiques són les seves pantalles AMOLED amb resolució de 2160 x 1200 píxels, un *framerate* de 90*fps* de fluïdesa d'imatge, un camp de visualització de 110 graus i el software *SteamVR* amb el que s'ofereix el sistema d'experiències virtuals *room-scale*. Abans d'instal·lar *HTC Vive* s'ha de verificar que es compleixen un requisits previs pel funcionament del sistema, el producte opera amb un ordinador que necessita unes especificacions tècniques mínimes i es necessita una espai de joc de mínim 2m x 1,5m. Per definir el dispositiu al complet s'han creat els apartats de hardware, software, funcionament del sistema i instal·lació del dispositiu per a conèixer més a fons quina és la tecnologia de *HTC Vive* i quins requeriments previs es necessiten per poder instal·lar el producte .

## 5.2 El Hardware

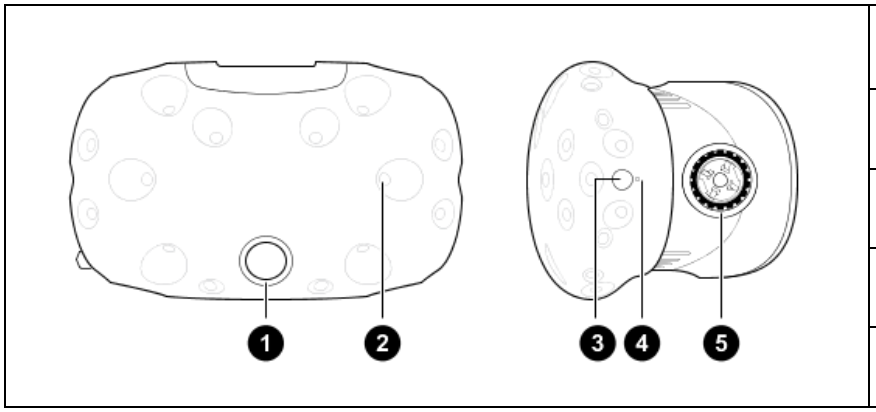
*HTC Vive* tecnològicament parlant és molt complert i complex ja que intervenen molts sistemes en el seu funcionament. *HTC Vive* té tres dispositius principals (el *Headset*, els *Controllers* i les *Base stations*) que realitzen tasques individualment i es transmeten informació entre ells per obtenir el procés global. Hi han processos de posicionament mitjançant sensors, processos d'input de dades de l'usuari mitjançant botons i processos de visualització dels resultats amb les ulleres. *HTC* i *Valve* han tingut que pensar molt bé la millor manera d'optimitzar el disseny del seu sistema per reduir al màxim el cost dels components electrònics sense que el resultat final es vegi afectat.

### 5.2.1 Ulleres de *HTC Vive* (*Headset*)

El *Headset* és la finestra que connecta l'usuari amb l'entorn virtual, està compost per:

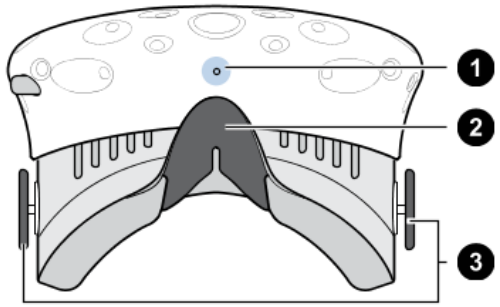
- Dues pantalles AMOLED de resolució 1080p que combinades tenen una resolució de 2160 x 1200 píxels amb un *framerate* de 90fps.
- Una càmera frontal manufacturada per la companyia *Sunny Optical Technology*.
- Un micròfon a la seva part inferior.
- Quatre tipus de sensors de posicionament (sensor de proximitat, giroscopis, acceleròmetres i posicionament de làser mitjançant un total de 37 fotodíodes).
- Un port HDMI , port àudio jack 3.5mm, dos ports *USB3.0* i un port d'alimentació *DC*.

Es poden veure les ubicacions dels components a la *Figura 5.2*, la *Figura 5.3* i la *Figura 5.4*.



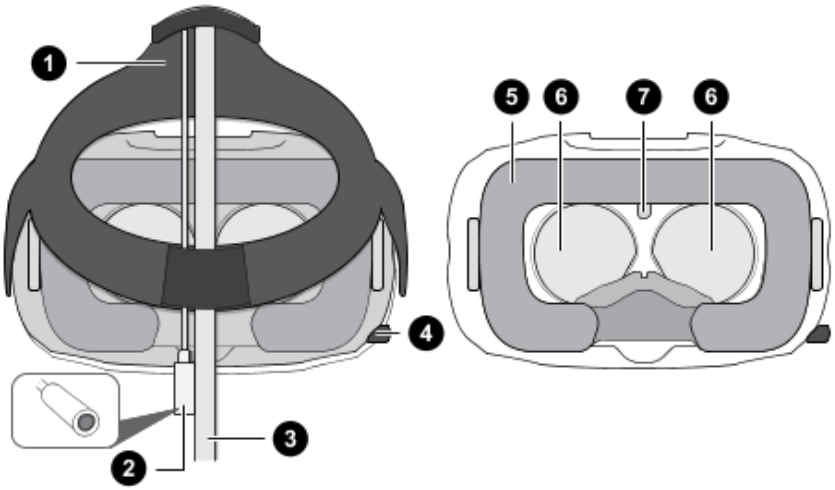

| | |
|---|---|
| 1 | Càmera |
| 2 | Fotodíodes |
| 3 | Botó del headset |
| 4 | Llum de estatus |
| 5 | Ajustador de lents |

*Figura 5.2: Esquema de components del headset, vista frontal i vista de perfil.*




| | |
|---|---|
| 1 | Micròfon |
| 2 | Coixí pel nas |
| 3 | Ajustador de lents |

*Figura 5.3: Esquema de components del headset, vista de planta.*




| | |
|---|---|
| 1 | Corretja |
| 2 | Àudio jack 3.5mm |
| 3 | Connector de ports 3 en 1 (2 ports USB 3.0 i alimentació DC). |
| 4 | IPD (ajustador de distància entre lents) |
| 5 | Coixí per la cara |
| 6 | Lents |
| 7 | Sensor de proximitat |

*Figura 5.4: Esquema de components del headset, vista posterior.*

El *Headset* té dos reguladors per ajustar l'efecte estereoscòpic a través de les lents i així aconseguir la màxima sensació d'immersió per a cada persona, en total pesa 0,555Kg. El recobriment de les ulleres és de plàstic dur amb una forma poc convencional, aquesta forma no està dissenyada per qüestions estètiques del producte sinó que s'adapta a la característica més important del component, la bona precisió de *tracking* del dispositiu. El *tracking* és una de les funcions essencials que té el sistema de *HTC Vive*, per això el primer disseny que es va realitzar a la fase de desenvolupament va ser el de la posició on anirien els fotodíodes, així es té el millor rastreig de posicionament del casc possible. Després es va adaptar la forma del *Headset* a l'estructura de fotodíodes que ja s'havia determinat.

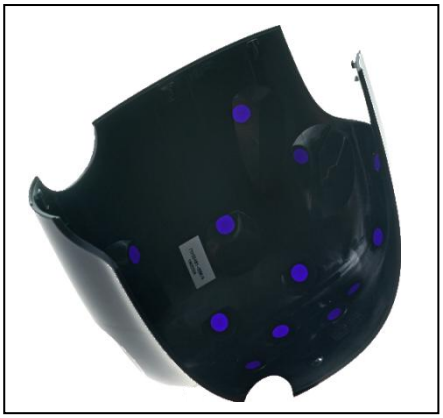


*Figura 5.5: Recobriment de plàstic del headset amb els filtres IR per protegir els díodes.*

Cada un dels fotodíodes de les ulleres va cobert per un filtre anomenat filtre *IR* o filtre d'infrarojos. Aquests filtres tenen la funció de reflectir i bloquejar les longituds d'ona infraroges i d'aquesta manera s'evita que apareguin interferències a l'hora de rebre les dades en el sistema. A la *Figura 5.5* es mostra el recobriment de les ulleres de realitat virtual, els punts violetes són els filtres IR i a l'unir el recobriment amb la resta cada filtre cobreix un fotodíode diferent.

*Figura 5.6: Imatge d'una pantalla del Headset amb la peça de plàstic que la suporta.*

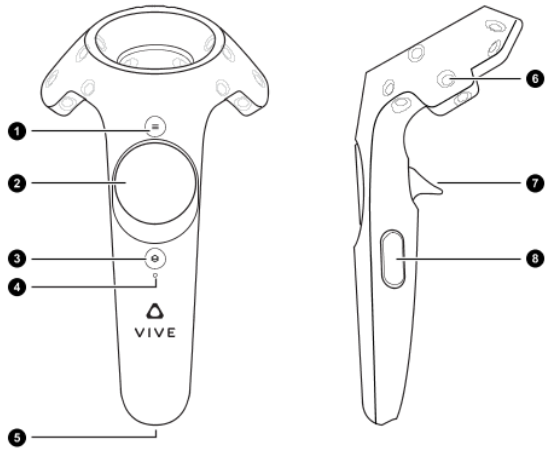
Les dues pantalles de visualització del *HMD* són del tipus *AMOLED* i han sigut manufacturades pel conglomerat electrònic d'empreses multinacionals *Samsung*. Aquestes pantalles es solen utilitzar en dispositius mòbils, són molt lleugeres i tenen un consum molt baix de potència.

*Figura 5.7: Imatge d'una lent.*

Les lents del *Headset* també són una part molt important del dispositiu. Gràcies a la forma convexa de les lents es focalitza la direcció de la vista a les pantalles de tal manera que es pot dur a terme l'efecte estereoscòpic. A més les lents tenen un patró de circumferències concèntriques per ajudar a enfocar els ulls al centre de les pantalles.

### 5.2.2 Controladors de *HTC Vive* (*Controllers*)

S'utilitzen dos *Controllers* per interactuar amb els objectes del món virtual, són els dispositius que connecten a l'usuari amb el món virtual. Al igual que el *Headset* els controladors tenen sensors basats en díodes fotovoltaics per obtenir la seva traçabilitat dins de l'espai de joc.




| | |
|---|---|
| 1 | Botó de menú |
| 2 | Trackpad |
| 3 | Botó del sistema |
| 4 | Llum d'estat |
| 5 | Port Micro-USB |
| 6 | Fotodíodes |
| 7 | Trigger |
| 8 | Grip button |

*Figura 5.8: Esquema de components dels Controllers, vista de planta i de perfil.*

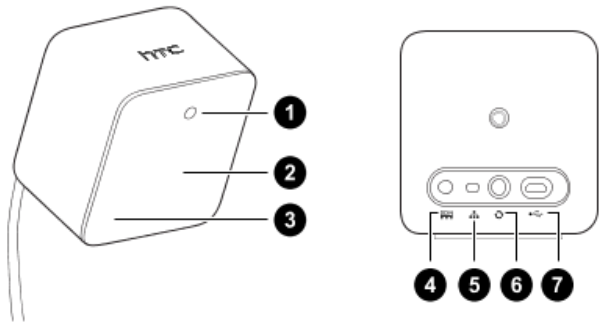
El disseny exterior de la forma dels controladors és una de les últimes fases que *HTC* i *Valve* van establir, en el procés de desenvolupament es tenien com a prototips dos controladors amb una antena receptora per calibrar el posicionament dels aparells, finalment van poder idear un disseny més estètic i va sortir l'estructura final en forma d'anella, que és on hi han els fotodíodes.

*Figura 5.9: Visualització dels fotodíodes del controlador al treure la coberta de l'anella.*

Han sortit la notícies de que Valve ja està treballant en una nova versió dels controladors.

### 5.2.3 Dispositius de posicionament (*Base stations*)

Els *Base stations* o *Lighthouses* són les dues caixes que creen tot l'espai de *tracking* pels fotodíodes a través d'un camp de làsers, són el far que guia a les ulleres i als controladors.




| | |
|---|---|
| 1 | Llum d'estat |
| 2 | Panell frontal |
| 3 | Indicador de canal |
| 4 | Port d'alimentació DC |
| 5 | Botó de canals |
| 6 | Port de sincronització de cable |
| 7 | Port Micro-USB |

*Figura 5.10: Esquema de components de les Base stations.*

Com es veurà més endavant quan es descrigui el tipus de sistema de funcionament del *HTC Vive* bàsicament les dues *Base stations* s'encarreguen de crear les senyals pels receptors i no realitzen cap procés d'anàlisi de dades, tot es tracta en calcular el temps en el que els sensors reben la senyal emesa. El port *Micro-USB* que porten incorporat serveix per poder actualitzar els aparells si fes falta.

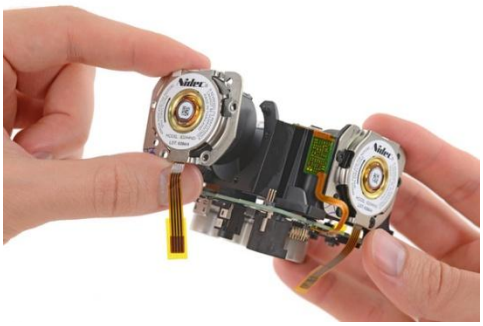


*Figura 5.11: Imatge dels motors dels Lighthouses.*

Encara que amb un primer cop d'ull no ho sembli, les *Base stations* són uns dispositius amb parts dinàmiques durant el seu estat operatiu, per realitzar l'entramat de camp de làsers els aparells tenen dos motors manufacturats per la companyia *Nidec* que giren a un certs *rpm* (revolucions per minut) hi cada un té un làser col·locat.

Igual que els controladors, han sortit notícies on s'informa que Valve ja té dissenyada una nova versió de les *Base stations* per treure al mercat en un futur.

### 5.2.4 Especificacions tècniques de funcionament

*HTC Vive* funciona amb un ordinador que processa les aplicacions i el sistema virtual treballa amb una qualitat gràfica mínima per poder fer córrer les escenes. Conseqüentment alguns ordinadors no poden suportar el sistema degut les targetes gràfiques, processadors o les memòries RAM. A part de la capacitat de processament, també es necessiten uns ports específics per poder connectar els aparells que s'utilitzen. A la *Taula 5.1* estan anotades les especificacions tècniques mínimes per poder utilitzar el sistema de realitat virtual.

| | |
|---|---|
| Processador | Intel Core i5-4590/AMD FX™ 8350, equivalent o superior. |
| Targeta gràfica | NVIDIA GeForce GTX 1060/AMD Radeon RX 480, equivalent o superior. |
| Memòria | 4 GB de memòria RAM. |
| Vídeo Output | 1 x port HDMI 1.4/DisplayPort 1.2 o superior. |
| USB | 1 x port USB 2.0 o superior. |
| Sistema operatiu | Windows 7 SP1, Windows 8.1 o Windows 10. |

*Taula 5.1: Especificacions tècniques pel funcionament de HTC Vive.*

Per comoditat de l'usuari Valve ha creat directament un programa que determina si l'ordinador que es vol utilitzar és apte per operar *HTC Vive*, el programa es diu *SteamVR Performance Test i* es pot trobar a la pàgina web de la plataforma de videojocs *Steam*. A més de les especificacions de l'ordinador també s'han de tenir en compte les alimentacions dels aparells (un mínim de cinc endolls convencionals disponibles, dos per les *Base stations*, un pel *Headset* i dos per carregar els *Controllers*) i s'ha de tenir en compte la zona disponible de joc (una zona de dimensions mínimes de 2m x 1,5m i un màxim de distància entre les *Base stations* de 5m).

## 5.3 El Software

El software que s'utilitza per córrer *HTC Vive* es diu *SteamVR* i prové d'una plataforma digital de distribució de software i hardware anomenada *Steam* creada justament per l'empresa *Valve. Steam* és una plataforma que utilitzen tan desenvolupadors independents com grans corporacions de software per la distribució dels seus videojocs i material multimèdia, de moment només està disponible per ordinadors. A part d'estar centrada a la distribució de contingut, *Steam* també és una xarxa social de comunicació entre persones i té una gran comunitat al darrere, també és una eina per administrar les actualitzacions dels videojocs d'ordinador i a més, *Valve* utilitza aquesta plataforma per vendre els seus productes (entre ells les ulleres *HTC Vive*). *E*l software *SteamVR* també està dissenyat per *Valve* i està creat específicament per fer córrer sistemes de realitat virtual. Es podria dir que *SteamVR* és el sistema central que controla tots els dispositius i totes les dades que reben els aparells per després processar els resultats, l'aplicació està disponible tan per *HTC Vive* com per *Oculus Rift*.

### *5.3.1* El software *SteamVR*

*SteamVR* serveix com a sistema d'iniciació i control de *HTC Vive*, l'aplicació prepara tots els requisits necessaris i deixa a punt el dispositiu pel seu funcionament. *HTC Vive* interactua amb l'ordinador gràcies a *SteamVR* que realitza un seguiment i control de tots els components del sistema (*Headset*, *Controllers* i *Base Stations*), mira si estan activats i si detecta algun problema obra un assistent d'ajuda per l'usuari. A més, el software realitza la instal·lació de tots els drivers (controladors de dispositius que té un ordinador) i comprova que estiguin a la última versió, també és el software amb el que es realitza la posada a punt prèvia de la zona de joc on s'interactuarà amb *HTC Vive*. A la *Figura 5.12* es mostra la interfície d'usuari que apareix quan s'engega *SteamVR*.

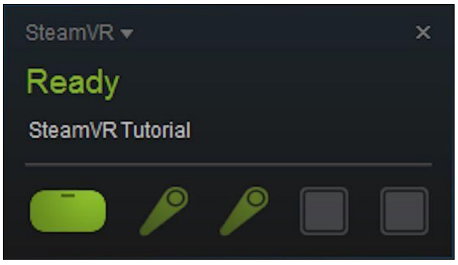



*Figura 5.12: Imatge de la interfície d'usuari de SteamVR.*

## 5.4 La tecnologia darrere del sistema *HTC Vive*

*HTC Vive* funciona amb la coordinació de tres tipus de sistemes tecnològics. Té dos sistemes principals (el de la visió tridimensional de les ulleres mitjançant un efecte estereoscòpic i el de posicionament espacial realitzant una tècnica de *tracking*) i un sistema secundari (un sistema de seguretat per saber els límits de la zona de joc).

### 5.4.1 Sistema de visió estereoscòpica

El sistema de visió estereoscòpica de les ulleres de realitat virtuals *HTC Vive* funciona amb el mateix principi que l'aparell de visió estereoscòpica inventat per Sir David Brewster al segle *XIX*. A través de dues lents convexes que desvien la direcció de visió de cada ull s'enfoca a dues pantalles que emeten imatges lleugerament desplaçades l'una de l'altra, el cervell uneix les dues imatges i percep com a resultat del procés una imatge tridimensional. Per obtenir una millor sensació estereoscòpica, s'han creat un reguladors per adaptar el dispositiu a la forma de la cara de cada persona.

La variació de la distància entre pupiles pot afectar a la visió tridimensional que s'aprecia ja que les ulleres treballen amb unes distàncies de precisió de l'ordre de mil·límetres, per això, en el casc s'ha inclòs un ajustador de distància interpupilar *IPD*, *Figura* 5.3.

*Figura 5.13: Imatge de les lents i l'ajustador IPD.*

*Figura 5.14: Imatge de l'ajustador entre lents i ulls.*

A més de l'estereoscòpia, una de les característiques principals d'un sistema d'immersió virtual és el camp de visió que s'obté amb els *HMD*. En el cas de les *HTC Vive* s'especifica que el camp de visió és de 110 graus però aquest número varia (en poca mesura) en funció de la distància entre les lents i els ulls, per aquest motiu també s'ha afegit un regulador de la distància entre ulls i lents per comoditat de l'usuari. *Figura 5.14*

### 5.4.2 Sistema de tracking *Lighthouse*

La idea principal darrere de la tecnologia de seguiment de les ulleres i els controladors és senzilla d'entendre però difícil d'aplicar, el posicionament es realitza de la següent forma: Simplement amb les *Base Stations* s'inunda l'habitació de llum no visible mitjançant làsers per crear un camp de referència pel *Headset* i els *Controllers (Figura 5.13)*. Les *Base Stations* fan la funció de far i les ulleres i els controladors amb els sensors que tenen repartits per tota la superfície reben la senyal, processen les dades i es posicionen a l'espai.

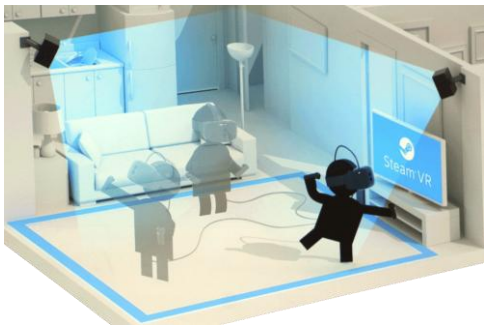



*Figura 5.15: Imatge representativa del camp de làsers que creen els Lighthouses.*

La tecnologia de *tracking* permet realitzar el posicionament amb dos *Base Stations* relativament barates i un total de 85 fotodíodes receptors de les senyals emeses. Els *Lighthouses* no tenen càmeres ja que no necessiten "veure" res, l'únic que fan és "il·luminar" la zona perquè els dispositius de moviment s'ubiquin en un sistema de referència. Les *Base Stations* realitzen dues accions, la primera la d'emetre feixos de llum a partir de làsers per tot el volum de joc, per tenir una referència espacial, i la segona la d'emetre flaixos periòdicament (mitjançant un conjunt de Leds) per tenir una referència temporal. *Figura 5.16*.

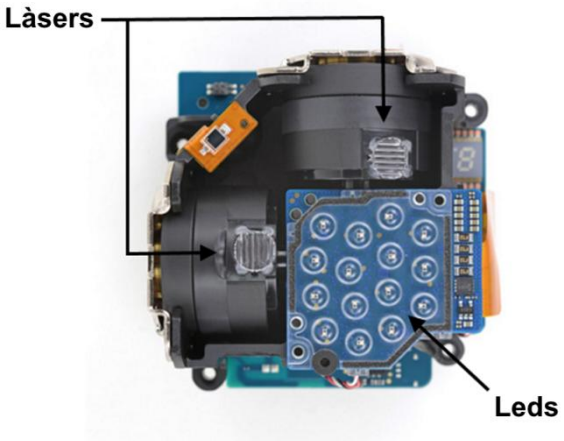



*Figura 5.16: Imatge de l'interior d'un Lighthouse.*

El camp de posicionament es crea amb dos rotors que tenen incorporats els làsers que emeten els feixos de llum. Els dos rotors giren a 3600rpm respecte els seus propis eixos (seixanta vegades per segon). Un rotor gira respecte l'eix vertical del sistema de coordenades de la *Base Station* (de baix a dalt) i l'altre respecte l'eix horitzontal (d'esquerra a dreta) i així s'abraça a tot el volum d'interacció.

A més dels làsers, la tecnologia de posicionament *Lighthouse* aprofita la periodicitat dels flaixos dels Leds per crear un sistema temporal d'emissió i recepció d'informació. Quan s'emet un flaix, el *Headset* i els *Controllers* comencen a contar el temps fins que un dels seus fotodíodes és tocat per un feix de llum d'un làser, i en aquell moment, es processa la relació entre el fotodíode que ha rebut el feix de llum i el temps que ha passat des del flaix per així saber la distància que ha recorregut el flaix abans d'arribar al sensori tenir la ubicació d'un punt del dispositiu a l'espai (ja que es sap la posició exacta de cada fotodíode). A la que més d'un fotodíode realitza aquest procés un algoritme calcula matemàticament en quina posició exacte de l'habitació està ubicat i orientat el dispositiu respecte les *Base Stations*.

### 5.4.3 Sistema de mallat *Chaperone*

El sistema *Chaperone* és el detector i limitador de la zona real de joc. Una vegada s'encén *HTC Vive* el sistema rastreja en tot moment on està l'usuari respecte les limitacions físiques de l'escenari i renderitza una malla blava de l'àrea disponible d'interacció *(Figura 5.17)* si el jugador s'apropa massa als extrems, amb aquesta tasca es dona la confiança a l'usuari de que no tindrà cap accident amb els objectes del seu voltant. El sistema obra les portes a tenir una finestra d'interacció entre el món virtual del *Headset* i el món real de l'entorn de joc ja que realitza el processament amb una càmera frontal.

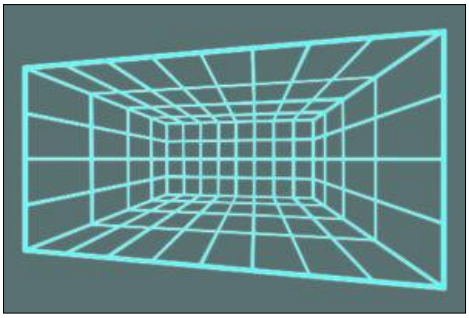


*Figura 5.17: Representació del ballat del sistema Chaperone.*

# 6. El software *Unity*, introducció a la creació de videojocs

*Unity* és l'eina essencial que s'ha utilitzat per crear les aplicacions d'aquest projecte, és un motor dissenyar per *Unity Technologies* pel desenvolupament de videojocs. En aquest apartat s'expliquen les principals característiques del programa i del seu editor, es fa una breu explicació de com funcionen els scripts en els videojocs i s'explica com es treballa a dins de l'editor amb el plug-in *SteamVR* proporcionat per *Valve* (no confondre amb el software que controla el dispositiu *HTC Vive*, el plug-in és una eina que ajuda a crear aplicacions de *HTC Vive* a dins de *Unity*).


*Figura 6.1: Logotip del motor Unity.*

## 6.1 Què és *Unity*

*Unity* és un motor de videojocs i programa d'ordinador que té una interfície d'usuari pel desenvolupament de software, és una eina que facilita la creació d'escenes virtuals mitjançant un editor que porta incorporat. Aquest motor bé amb les seves pròpies físiques, il·luminació, modelatge d'escenes i tota mena d'eines d'edició, pel que comparant-lo amb altres motors de desenvolupament és molt més fàcil endinsar-s'hi si no es tenen coneixements complexos de programació i de creació d'aplicacions. L'empresa *Unity Technologies* ha dissenyat una interfície d'usuari del software agradable i intuïtiva perquè d'aquesta manera treballar amb el sistema sigui més fàcil i còmode, a més, l'empresa proporciona una comunitat online on hi han tutorials d'aprenentatge i una llibreria molt completa amb les descripcions de totes les classes que es poden programar via scripts. Una altra de les principals avantatges del programa és que hi ha una tenda virtual anomenada *Asset Store* on es poden trobar modelats, textures i escenes fetes que es poden importar a les creacions que un mateix desenvolupi, ja sigui descarregant paquet gratuïts o de pagament. Es diu que *Unity* és una eina de suport multi plataforma ja que deixa exportar les aplicacions amb el format de plataforma que s'esculli, ja siguin sistemes operatius d'ordinador (*Linux, Mac, Windows*), plataformes mòbils (*iOS, Android, Windows Phone*...), consoles (*Playstation4, Xbox One, WiiU, 3DS, Nintendo Switch*) o plataformes de realitat virtual (*SteamVR, Oculus Rift, Google Cardboard...*).

*Unity* està compost de tres estructures principals:

- El motor.
- L'editor.
- Els mòduls d'exportació a les plataformes.

### 6.1.1 El motor

El motor de *Unity* és el "motor del videojoc", qualsevol creació que es realitzi amb *Unity* funciona gràcies al motor del software, es podria dir que és la base de la qual es regeix el sistema. El motor crea les lleis de l'univers virtual (una de les característiques més útils del software) a través de simulacions i deixa que els desenvolupadors omplin les escenes amb les seves creacions i scripts.

### 6.1.2 L'editor

L'editor de *Unity* és la part més tangible del software, és on els creadors passen la major part del seu temps treballant amb el programa, el jugador final no intervé en aquest procés i en la majoria dels casos si no es coneix l'eina no es sap ni que existeix. A l'editor es compacten totes les interfícies, les finestres de treball, les eines i els menús que serveixen pel desenvolupament. L'editor inclou tres interfícies: la zona visual on es creen les escenes i els entorns, la zona on s'escullen i determinen les propietats del objectes i la zona on es programen els scripts (que és un programa complementari, el programa predeterminat és el *MonoDevelop* però es poden utilitzar altres com el *Visual Studio*). Més endavant s'aprofundeix més amb l'editor i les seves característiques.

### 6.1.3 Els mòduls de distribució

L'exportació de *Unity* es realitza amb una eina de distribució multi plataforma, el principi en han volgut materialitzar els seus creadors és el de "desenvolupa una vegada i exporta a les plataformes que vulguis". Això suposa una avantatge molt gran pels desenvolupadors ja que poden crear una aplicació i amb uns canvis mínims distribuir-la a més d'una plataforma o sistema. Els mòduls de distribució de *Unity* s'encarreguen d'empaquetar la informació de la forma correcte per cada tipus de sistema diferent al que va adreçat.

## *6.2* Estructura de l'editor de *Unity*

L'editor de *Unity* s'ha intentat dissenyar d'una manera en que totes les accions del sistema es puguin fer arrastrant i enllaçant els objectes que es construeixin, aquest mètode funciona fins i tot amb els scripts, l'assignació de variables o la creació de modelats. La interfície de treball de l'editor de *Unity* està basada en panells on cada un té una funció diferent, la interfície es pot modificar a gust de l'usuari però sol estar estructurada sempre amb cinc finestres bàsiques, la *Scene View*, el *Hierarchy*, la *Game View*, el *Project* i l'*Inspector*. A la *Figura 6.2* es mostra la interfície d'usuari i a continuació s'explica la funció de cada panell.

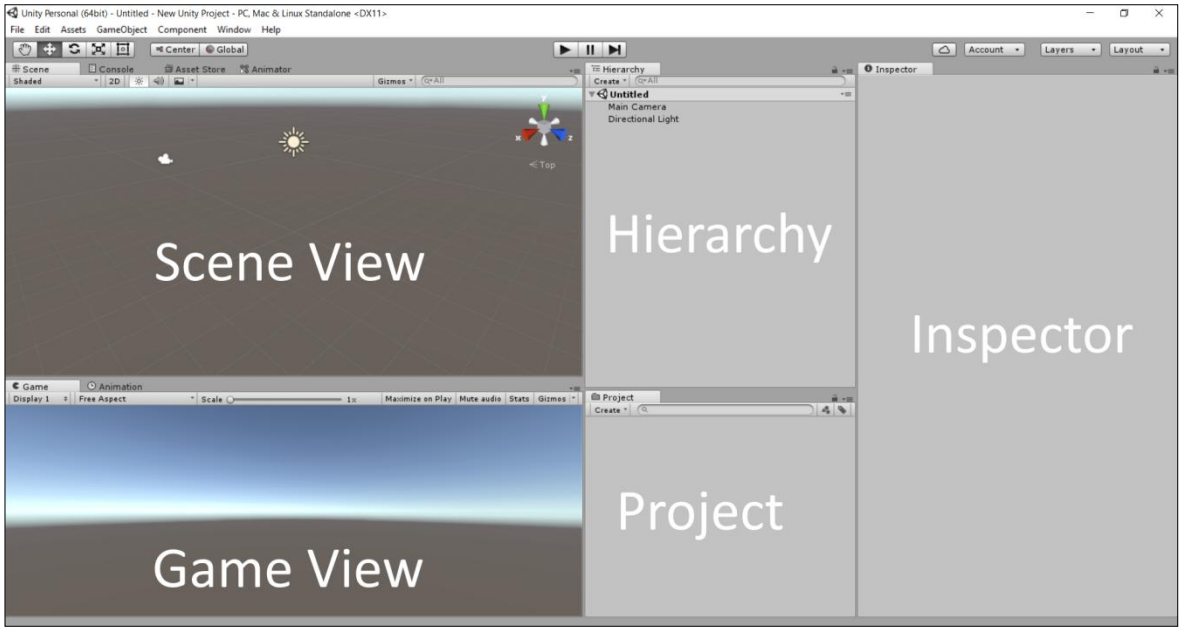



*Figura 6.2: Interfície d'usuari de l'editor de Unity.*

- La *Scene View*: És la pantalla on visualment es construeix el videojoc, afegint i manipulant els objectes en 2D i 3D, les càmeres, la il·luminació i bàsicament tot el que interactua amb l'escena. Gràcies a la *Scene View* es pot tenir una idea de com està muntat l'entorn virtual ja que es mostren tots els objectes de dins de l'escena.

- El *Hierarchy*: S'utilitza per localitzar els objectes que hi han a dins de la escena i serveix per seleccionar-los d'una forma ràpida, és un llistat dels nomes del objectes on també apareixen les jerarquies de cada un en el cas de que estiguin emparentats (agrupats). És molt important tenir un *Hierarchy* ben organitzat a l'hora de realitzar un videojoc per tenir una estructura simple on es puguin trobar ràpid les coses.

- El panell *Project*: Els projectes de *Unity* estan emmagatzemats a l'ordinador en forma carpetes on cada una guarda diferent tipus d'informació de la aplicació, al panell *Project* es visualitza l'informació d'una de les carpetes, la anomenada *Assets*. A la carpeta *Assets* és on es guarden totes les dades que s'utilitzen per la creació d'objectes i escenes de l'editor. La carpeta *Assets* guarda bàsicament tot el material que s'utilitza en el projecte, escenes, scripts, materials, imatges, prefabs etc.

- L'*Inspector*: Proporciona el detall dels components de l'objecte que està seleccionat en aquell moment, des d'aquest panell es poden modificar les seves propietats com ara el tamany i la posició i adjuntar-li més components com ara físiques o scripts.

- La *Game View*: És el panell on es veu la pantalla de joc que es mostrarà al producte final, permet provar el videojoc sense tenir que sortir de l'editor i veure si la visualització és l'adequada.

## 6.3 Conceptes bàsics de funcionament

### 6.3.1 *GameObjects* i components

*Unity* funciona amb les seves pròpies regles i processos de comunicació entre els objectes que interactuen a les escenes. Per poder crear entorns *2D*/*3D* amb físiques, renderitzats d'imatges o creació d'interfícies d'usuari entre moltes altres funcions els creadors van dissenyar les bases internes de l'aplicació amb una estructura de propietats molt ben definida, aquestes propietats ajuden a l'usuari a modificar de forma simple característiques d'un objecte. Ja sigui el color, el tamany, el pes, la posició a l'espai o els scripts que porta un *GameObject*, tot està estructurat i ordenat de la mateixa manera en forma de components. Els *GameObjects* són els objectes s'afegeixen a les escenes, qualsevol cosa que estigui dins d'una escena és considerat un *GameObject*, aquests objectes estan estructurats per un número de components que són els que defineixen les propietats de cada un d'ells. Existeixen molts tipus de components a dins de *Unity*, com per exemple components físics, components de rendering, d'interfície d'usuari o scripts. A part de tenir una estructura més simple de comprensió, un dels avantatges de treballar amb components és que cada tipus de *GameObject* nou que es crea ve amb un set de components predeterminats associats. Depenent del tipus de *GameObject* que s'afegeixi es tindrà un set de components o un altre, d'aquesta manera cada tipus d'objecte ja ve caracteritzat amb les seves propietats bàsiques.

Qualsevol objecte dins de l'escena té com a mínim un component predeterminat anomenat *Transform,* que ajuda a posicionar i a dimensionar al *GameObject* a l'espai virtual amb una posició, una rotació i una escala que es determinen d'acord als eixos x,y,z de l'espai de coordenades de l'escena. Després, depenent del tipus d'objecte que sigui, tindrà una combinació de propietats o una altra. A la *Figura 6.3* es pot veure en el panell de l'*Inspector* un exemple dels components d'un tipus objecte, en aquest cas l'objecte té quatre components associats, el predeterminat de *Transform*, dos components de *meshing* anomenats *Mesh Filter* i *Mesh Renderer*, un component de físiques anomenat *Collider* i un component de rendering anomenat *Material*. Aquest set que es veu a la imatge és el predeterminat per objectes sòlids. Es poden afegir tot tipus de components a un objecte per fer-lo tot el complex que es desitgi. A la comunitat de *Unity* hi han llibreries on s'expliquen tots els components que té el programa i les seves funcions.

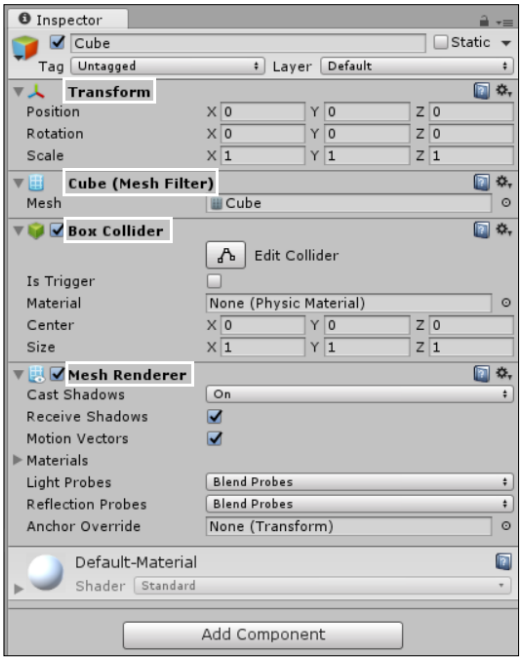



*Figura 6.3: Panell Inspector amb components d'un GameObject tipus sòlid.*

Per l'enteniment d'alguns conceptes tècnics del projecte és necessari explicar els components bàsics amb els que normalment es treballa en qualsevol projecte de *Unity*, es diferenciaran entre components de rendering, components d'aparença, components de físiques, components d'interfície d'usuari i scripts.

- Components de rendering:

  Camera: És el dispositiu a través del qual l'usuari final visualitza l'entorn virtual, el que es veu per la càmera és el que es veurà a la pantalla del jugador.

  Skybox: Els *skyboxes* són la imatge que envolten els límits de l'escena, ensenyen com està representat el món més enllà de l'entorn d'interacció i són una eina de suport per donar credibilitat a l'escena. Estan compostos de sis imatges que el programa processa per adaptar-les a la zona virtual. Cada imatge és col·locada a una vista en concret de l'espai tridimensional.

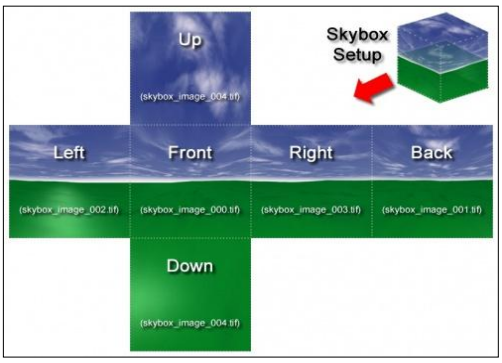



*Figura 6.4: Esquema de les parts d'un Skybox.*

  Material (Shader): És la textura del material, controla els colors i les reflexions de l'objecte, s'utilitza per posar-li al material textures o colors. El *shader* és un tipus de script que calcula matemàticament el color de cada píxel de l'objecte basant-se amb la llum rebuda i el tipus de material que s'ha definit prèviament.

- Components de posicionament:

  Transform: És el component que situa a l'objecte a dins de l'espai virtual de l'escena, controla la seva posició, la seva rotació i la seva escala (dimensions).

- Components físiques:

  Rigidbody: Permet a l'objecte estar regit pel motor de físiques de *Unity*. El *Rigidbody* actua en base a forces i moments de torsió per moure els cossos de manera realista.

Qualsevol objecte ha de tenir un *Rigidbody* per estar regit per la gravetat, estar influenciat per forces creades via scripts o interactuar físicament amb altres objectes.

Colliders: És la zona volumètrica de l'objecte que interactua amb les lleis físiques de l'entorn, tot objecte ha de tenir un *Collider* si es vol que estigui influenciat per col·lisions i forces externes. *Unity* proporciona una gama limitada de formes geomètriques pels *Colliders*, poden ser *Box Colliders* (cubs), *Sphere Colliders* (esferes) o *Capsule Colliders* (capsules). S'ha d'intentar assimilar els *Colliders* el màxim possible als sòlids que representen, a la

Figura 6.5 es mostra l'exemple d'un cub amb el seu *Collider* representat per línies verdes (el *Collider* s'ha desplaçat expressament del centre del cub per la seva visualització).

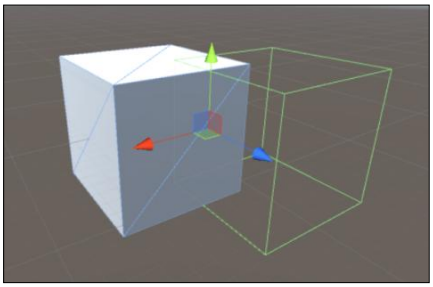


*Figura 6.5: Imatge d'un cub amb el seu Collider.*

- Components d'interfície d'usuari:

  Totes les interfícies d'usuari van acompanyades d'un tipus d'eina anomenada *Canvas*, el *Canvas* representa l'espai on la UI es mostrarà. Tenir un *Canvas* ajuda a posar a més d'una UI (ja sigui imatge, text o botó) en una mateixa zona de forma conjunta.

  Text: Mostra un text dins del *Canvas*.

  Image: Mostra una imatge a dins de l'objecte.

- Scripts: Els scripts també actuen com a components de l'objecte, si es vol assignar algun objecte en concret a una variable de l'script es pot fer directament mitjançant el panell *Inspector*. També es poden modificar el valor de les variables si es programa de la forma adequada per no tenir que obrir l'editor de scripts.

Prefabs:

Una altra funció essencial a l'entorn de l'editor de *Unity* són els *Prefabs* (*GameObjects* prefabricats). Els *Prefabs* van molt bé quan es té una gran quantitat d'objectes semblants a la escena, si es canvia el *Prefab* d'un objecte és modifiquen també tots els altres objectes amb el mateix *Prefab*. L'únic que fa falta per convertir un objecte en un *Prefab* és arrastrar-lo en el panell *Project* i després des del panell anar afegint aquest *Prefab* dins de l'escena les vegades que es necessiti. Una altra avantatge de treballar amb aquesta eina és que cada *Prefab* guarda els seus components interns, pel que si es té un *GameObject* amb unes característiques que es tornaran a utilitzar de la mateixa forma, només fa falta editar un cop els components del *GameObject*, convertir-lo en *Prefab* i es treballar amb aquesta sistemàtica durant tot el projecte, així es poden editar totes les propietats amb un sol click.

A la *Figura 6.6* es pot veure el panell *Hierarchy* d'un projecte de *Unity* amb diversos *Prefabs* (que es són els *GameObjects* de color blau), una altra avantatge de treballar amb *Prefabs* és que es poden utilitzar a totes les escenes que té un projecte. A més, es poden utilitzar els *Prefabs* per poder importar de forma ràpida grups d'objectes predissenyats per realitzar una funció específica a qualsevol dels projectes amb els que s'estigui treballant.


*Figura 6.6: Imatge del Hierarchy amb diversos Prefabs.*

### 6.3.2 Estructura dels scripts d'un videojoc de *Unity*

Ja s'ha vist que el comportament d'un *GameObject* va regit pels seus components intens, encara que els components que proporciona *Unity* siguin molt versàtils, s'haurà d'anar una mica més enllà per poder dissenyar un videojoc i és indispensable realitzar scripts. Realment els components de l'editor de *Unity* són scripts fets per la pròpia empresa per facilitar les tasques de desenvolupament, l'avantatge de realitzar scripts propis és que es poden adaptar a l'aplicació en concret que es dissenya i són molt més efectius per implementar les característiques específiques que es busquen en el sistema. *Unity* suporta dos llenguatges de programació, el *C#* i el *UnityScript* (una versió particular del *Java*), en aquest treball s'ha utilitzat el llenguatge C# per realitzar els scripts i és el que s'utilitzarà per les explicacions. Quan es programa amb el sistema de classes de *Unity* i es té algun tipus de dubte sobre la crida de les funcions, és molt recomanable anar a la pàgina web de *Unity*, allà hi ha un manual amb les descripcions dels tipus de funcions i exemples amb els mètodes de crida.

La programació es fa amb una aplicació externa, ja sigui *MonoDevelop* (el predeterminat) o una altra, i en el cas de voler veure el funcionament dels scripts (per veure algun valor d'alguna variable a temps real) es fa des del panell *Console* de l'editor, que també avisa quan hi ha algun error de programació (amb la ubicació de l'error i el perquè s'ha provocat).

```
using UnityEngine;
using System.Collections;

public class NomDelComponent : MonoBehaviour {

	// Utilitzar per l'inicialització.
	void Start () {

	}

	// L'Update és cridat una vegada per cada frame.
	void Update () {

	}
}
```

*Figura 6.7: Script estàndard que crea el propi sistema de Unity.*

Al crear un script des de zero el programa ja enllaça una estructura estàndard per la bona interacció amb el motor de *Unity*, s'ha de pensar dels scripts com a un component més del *GameObject* on va associat, el nom de l'script és el nom del component que es crea.

Automàticament el programa crida la classe bàsica de *Unity* anomenada *MonoBehaviour* que és des d'on es deriva qualsevol funció a dins del programa. La primera cosa a analitzar quan es veu per primera vegada l'estructura d'un script són les dues funcions que es creen predeterminades, la de *Start()* i la de *Update()*. *Unity* té una jerarquia de noms on hi han funcions que es processen abans que d'altres, en aquest cas si es posés codi a la funció *Start()* aquest es processaria només al iniciar-se l'aplicació i si es posés codi a la funció *Update()* aquest es processaria a cada frame (aniria bé per posar codis relacionats amb el moviment o inputs de botons per exemple), es pot visualitzar la jerarquia de noms de les funcions al document buscant *"Execution Order of Events"* al manual de *Unity*.

Una de les característiques més importants i principals dels scripts és que et deixen visualitzar i modificar els components dels objectes i les seves propietats a temps real, per exemple, un script simple per practicar seria el de visualitzar pel panell *Console* la posició en

l'eix x d'un objecte de l'escenari al prémer la tecla espai, tot es basa en començar des d'un nivell general i anar aprofundint fins a arribar al nivell amb el que vols interactuar. En aquest cas s'hauria d'utilitzar una funció de *Input* (relacionada amb tots els dispositius d'entrada a l'aplicació, ja sigui el teclat, el ratolí, un comandament etc.) amb la tasca de prémer el botó (es pot mirar a la llibreria de *Unity* a l'apartat *Input*, en aquest cas *GetKey*) i s'hauria d'arribar fins el nivell de posicions de l'objecte a l'eix x (*GameObject→Transform→Position→eix x*).

```
using UnityEngine;
using System.Collections;

public class NomDelComponent : MonoBehaviour {

    //Variables que es volen cridar.
    public GameObject objecte;

    // L'Update és cridat una vegada per cada frame.
    void Update ()
    {
        if (Input.GetKeyDown("space"))
        {

            print(objecte.transform.position.x);
        }
    }
}
```

Es crida a l'objecte amb una variable pública, això vol dir que s'haurà d'enllaçar aquest des de l'editor arrastrant-lo.

Si es pressiona la tecla espai es realitza el codi de dins dels {}.

S'"imprimeix" la posició de l'eix de les x de l'objecte a la consola.

*Figura 6.8: Script bàsic d'exemple.*

A la *Figura 6.8* es pot veure l'exemple passar a script, totes les zones de processos estan separades sempre per dos {}. El programa de creació del scripts t'ajuda a auto completar el codi amb diverses opcions amb el que es fa més fàcil la programació. Hi ha molt tipus de variables a part de la de *GameObject*, com per exemple, *int* (números enters), *float* (números decimals), *Tranform* (selecciona directament el component *Transform* del *GameObject* enllaçat), *Canvas,* entre molts altres.

Un script sempre ha anar enllaçat a un objecte de l'escena, hi ha diferents mètodes per cridar un objecte dins d'un script, es pot fer enllaçant-lo com a una variable pública (com a l'exemple) i a la *Figura 6.9*, o enllaçant l'script directament a l'objecte com a component i cridant-lo a dins del programa posant simplement *gameobject*.

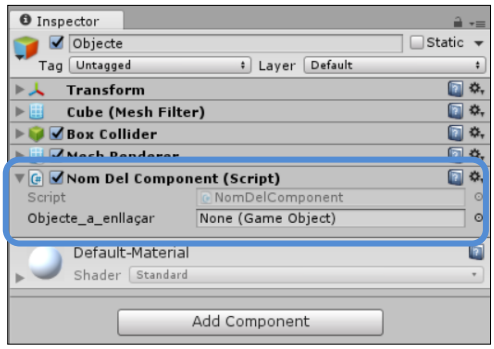



*Figura 6.9: Imatge d'un component script.*

A partir d'aquí si no es sap com realitzar una certa funció només fa falta veure exemples del que es vol realitzar i adaptar-ho de la millor manera al script que es realitzi, tot es basa en saber buscar la informació correcte en les llibreries des del buscador que la plataforma web de *Unity* porta incorporat.

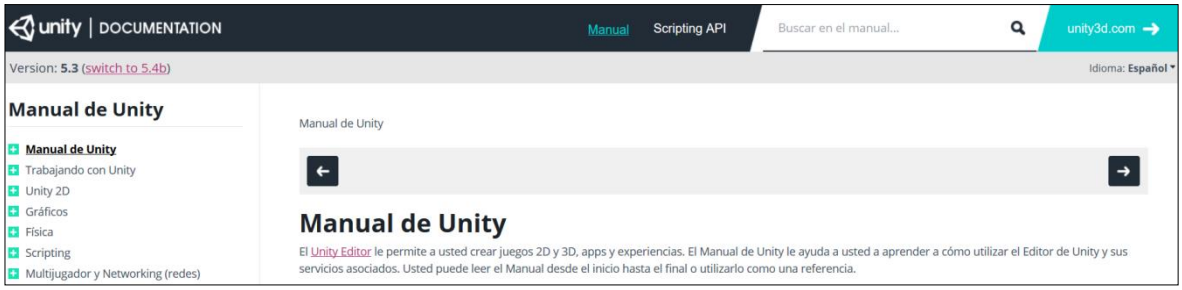



*Figura 6.10: Imatge extreta de la pàgina web de la llibreria de documentació de Unity.*

### 6.3.3 Influencia de les versions de *Unity* al projecte

És molt important treballar sempre un projecte amb la mateixa versió amb el que s'ha iniciat de *Unity*, hi ha funcions que s'expressen diferent depenent de les versions amb les que es treballa i si passes el projecte d'una versió a una altra sense realitzar canvis entremig, pot ser que els scripts es vegin afectats i no puguis recuperar el document fàcilment. No obstant, si es vol fer un canvi de versió a les pàgines de *Unity* normalment diuen els passos que s'han de realitzar per poder passar d'una a l'altra. Moltes vegades les actualitzacions del programa venen amb molts pocs canvis que no afecten en gran mesura amb l'experiència d'edició, pel que és recomanable no canviar la versió (quan es realitza una actualització de software sempre es pot mirar a la pàgina web de *Unity* quines millores comporta el canvi).

# 7. Inicialització a la programació de la realitat virtual, el plug-in *SteamVR*

## 7.1 Inicialització al paquet *SteamVR* de *Unity*

*SteamVR* és el plug-in amb el que es programa amb *Unity* pel conjunts del dispositius del sistema *HTC Vive*. Aquest paquet té totes les funcions essencials perquè el *Headset* i els *Controllers* interactuïn amb l'espai virtual de les escenes de l'editor de *Unity*. La característica més important que proporciona aquest plug-in són els scripts generals per poder comunicar-se amb els dispositius de *HTC Vive*, ja siguin les posicions, els inputs dels controladors, les dades de la zona de joc real o els renders de les imatges dels aparells, aquest plug-in tindrà un script amb el qual es podrà treballar la informació que es necessiti.

Els dos scripts principals que s'utilitzen per poder activar *HTC Vive* amb l'editor són:

- El ***SteamVR_Camera****:* Script que afegeix el *Headset* i *Controllers* a dins de l'escena.
- El ***SteamVR_TrackedObject***: Script que emmagatzema els moviments dels dispositius (*Headset* i *Controllers)*.

Hi han molts scripts a dins del plug-in que interactuen de vàries maneres diferents amb el sistema (*Figura 7.1*), però per sort, el paquet *SteamVR* bé preparat i inclou directament una carpeta de *Prefabs* on només fa falta arrastrar tres components per poder interactuar amb *HTC Vive*. La explicació del procediment s'explica al següent apartat.

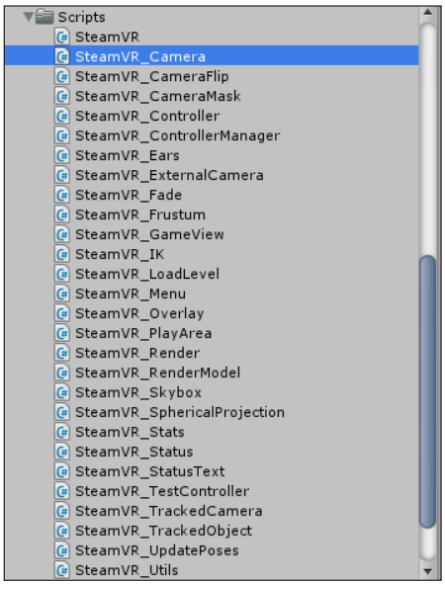



*Figura 7.1: Els scripts base del plug-in SteamVR.*

## 7.2 Descripció dels objectes de *SteamVR*

Primer de tot s'ha de descarregar el plug-in de *SteamVR* de la *Asset Store* de *Unity* (la *Asset Store* es pot trobar com a una finestra més de l'editor de *Unity* a la pestanya *Window*), per fer-ho, només fa falta posar-se al buscador de la *Asset Store* i escriure "*steamvr*". El plug-in vàlid és el creat per la companyia *Valve Corporation* amb nom complet *SteamVR Plugin*.

El plug-in *SteamVR* està organitzat amb dues carpetes principals, una anomenada *Plugins* i l'altra anomenada *SteamVR*. La carpeta *SteamVR* serà la principal a l'hora de treballar ja que hi han els útils que s'utilitzaran per importar assets al projecte, a dins de la carpeta n'hi han moltes altres, la informació essencial que es necessitarà està en les carpetes *Scripts* i *Prefabs*. El *Prefab* amb el que es treballarà a l'escena és el *CameraRig*.

- Carpeta ***Prefabs***: conté (CameraRig | Status | *SteamVR*)

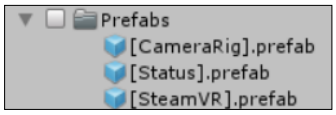



*Figura 7.2: Carpeta de Prefabs.*

*Prefab CameraRig*: El prefab *CameraRig* s'encarrega de que un cop iniciada l'aplicació es visualitzi en tot moment l'escena creada a *Unity* a través de les ulleres. Aquest prefab relaciona les posicions i rotacions (x,y,z) de *Unity* amb les posicions i rotacions (x,y,z) de les ulleres *HTC Vive* i els controladors. Aquesta orientació a l'espai s'actualitza a cada frame i "transforma" (mou) la posició i orientació del *Headset* i dels *Controllers* en tot moment per l'escena, d'aquesta manera, al fer el seguiment del jugador a la vegada que va transformant la seva perspectiva dins del joc, dona la sensació d'immersió a dins de l'entorn virtual.

A dins del prefab *CameraRig* hi han tres *GameObjects* vinculats, el *Controller left*, el *Controller right* i el *Camera head*, seran els objectes on es posaran els scripts propis:

Els ***Controllers (left/right)***: són els *GameObjects* dels controladors, té com a components predeterminats un script de tracking *TrackedObject* i un *Model* amb un script *RenderModel* del renderitzat de la imatge dels aparells.

A ***Camera (head)***: es tenen les ulleres *HTC Vive* vinculades amb el nom de *eyes* (la càmera) i amb el nom de *ears* (la sortida d'àudio).

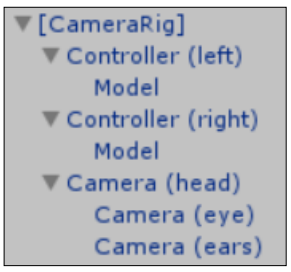



*Figura 7.3: Desglossament del Prefab CameraRig.*

*Prefab SteamVR*: És el *Prefab* es crea a temps real pel CameraRig quan es corre l'aplicació que s'està desenvolupant. que de controlar la informació de l'àrea de joc disponible. *Unity* recol·lecta les dades del sistema de joc del software *SteamVR* (aplicació que fa funcionar les *HTC Vive* amb l'ordinador). Aquest *Prefab* s'encarrega d'agafar aquestes dades i enllaçar-les a *Unity*, permet canviar l'experiència d'immersió virtual a assegut, dret o no especificat.

*Prefab Status:* El *Prefab* status s'encarrega de controlar les preferències de control de *SteamVR* com ara tipus de text, color de la *UI* etc. No fa falta modificar res perquè ja bé predeterminat.

## 7.3 Set-up de SteamVR a Unity

- **Preparar el plug-in per una escena nova de Unity:**

  - S'ha de descarregar *SteamVR* i arrastrar el *Prefab CameraRig* a dins de l'escena.
  - S'ha de desactivar la *Main Camera* predeterminada (s'utilitzarà la càmera del *Prefab Camera head*).
  - S'ha d'iniciar el software *SteamVR* i córrer l'escena, ja es podrà veure l'entorn virtual des de les ulleres. Ara només faria falta dissenyar els objectes de dins de l'escena i els scripts necessaris pel seu funcionament. Hauria d'haver quedat un *Hierarchy* com el de la *Figura 7.4.*

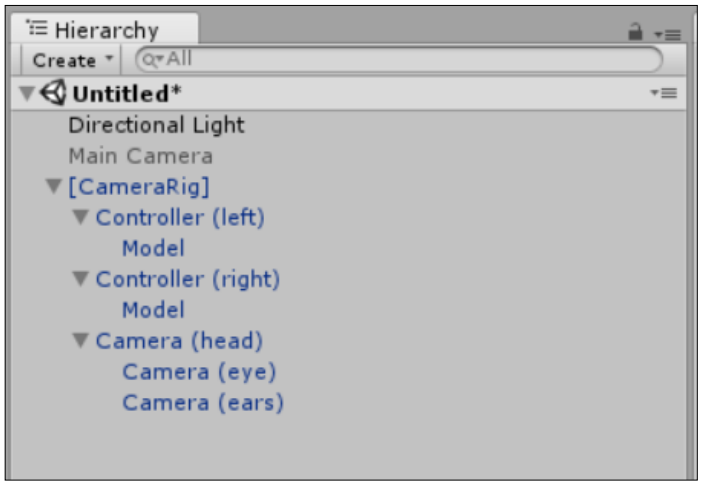



*Figura 7.4: Hierarchy amb totes les components essencials per iniciar una escena de realitat virtual amb les HTC Vive.*

- **Preparar el plug-in per una escena ja creada de Unity:**

  - S'ha d'anar a la *Main Camera* i afegir-li el script ***SteamVR_Camera***, apareixeran dos scripts nous com a components a de la *Main Camera* s'ha de clickar en un d'ells on hi ha un botó *Expand*, directament el programa crearà quatre components diferenciades de la càmera, el (*origin*), a dins de *origin* el *(head)* i a dins de (*head*) els *(eye)* i *(ears)*.


*Figura 7.5: Transformació de la Main Camera al accionar "Expand".*

El o*rigin* s'utilitza per calibrar la dimensions de la *play area*, el *head* per tenir el *tracking* de les ulleres, el *eye* per la càmera (té el script *SteamVR_Camera*, hi ha l'opció de prémer el botó *Collapse* per tornar a posar la *Main Camera* normal), i els *ears* tenen les opcions d'àudio. El *HMD* ja podria córrer l'escena virtual.

- El pròxim pas seria realitzar els *GameObjects* dels controladors. A dins de *origin* s'han de crear dos GameObjects buits (un per cada Controlador), es pot crear només un i després duplicar-lo per tenir l'altre, anomenarem al primer objecte *Right Controller.* A dins de *Right Controller* s'ha d'afegir l'script per rastrejar els controllers (afegir l'script ***SteamVR_TrackedObject*** com a component de l'objecte) i s'ha d'afegir el renderitzat dels controlador per veure'l en escena (crearem a dins de *Right Controller* un objecte anomenat *Model* i li afegirem el script ***SteamVR_RenderModel***, el render treballa amb un *shader* en concret, des de el component hi ha l'opció de canviar el *shader*, busquem el *shader standard*). Un cop creat el primer controlador s'haurà de duplicar el *Right Controller* i anomenar a la copia *Left Controller*.

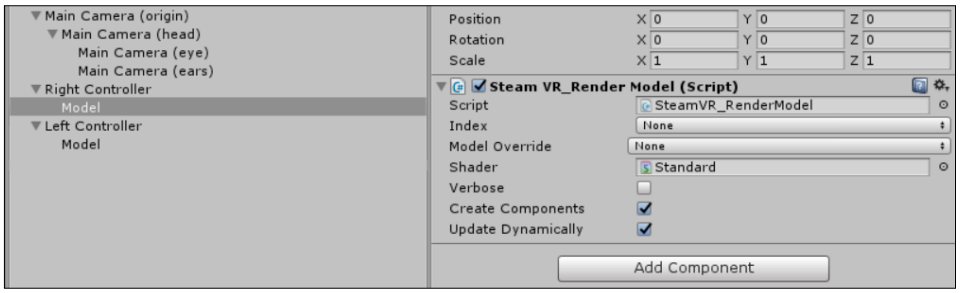



*Figura 7.6: Visualització de l'editor després d'haver construït els GameObjects dels dos controladors.*

- Com a pas final es necessita controlar la escena de joc i un sistema que manegi els dos controladors (per comprovar que els dos aparells estiguin activats a l'iniciar l'escena, avisar si hi ha algun error i tenir referència de quin és el controlador esquerra i dret).

  Aquest sistema serà el script **SteamVR_ControllerManager** que s'ha d'afegir com a component l'objecte (*origin*). S'han d'arrastrar els dos *Tranform* dels controllers Left i Right perquè sàpiga quin són cada un.

  Després també s'haurà d'afegir el script ***SteamVR_PlayArea***, que bàsicament deixarà controlar totes les propietats relacionades amb la zona de joc. És molt útil per poder veure les dimensions de l'àrea d'interacció a l'escena virtual de l'editor.

Si s'han seguit tots els passos anteriors ja es podrien córrer les escenes de realitat virtual (s'ha de recordar que, perquè funcioni el sistema, el software *SteamVR* per engegar el dispositiu *HTC Vive* ha d'estar actiu a l'ordinador).

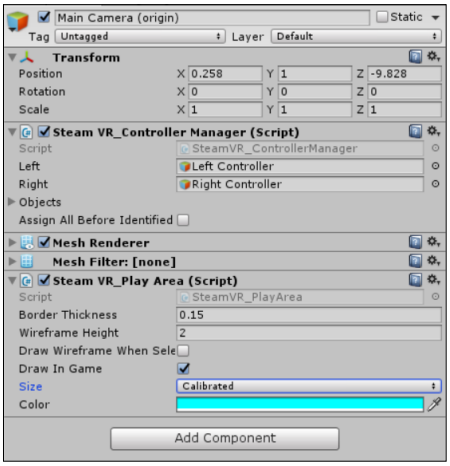



*Figura 7.7: Els components de Main Camera (origin).*

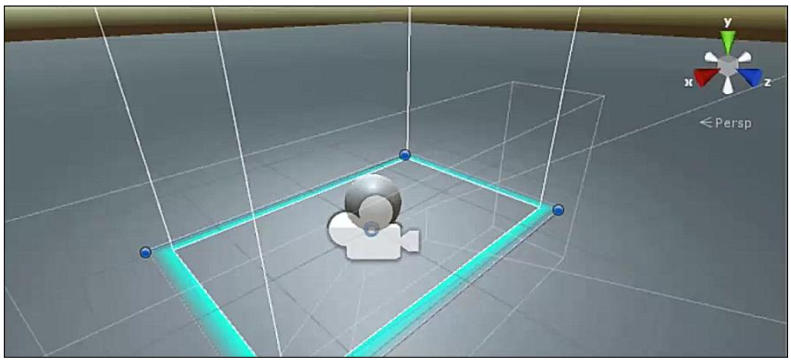



*Figura 7.8: Visualització dels límits de l'àrea de joc a dins de l'editor de Unity.*

ETSEIB

# 8. Disseny i construcció de les aplicacions de suport

Per practicar amb el programa *Unity* i veure les seves funcionalitats i estructura, abans de crear el *serious game* de rehabilitació, s'han realitzat dues aplicacions que han ajudat com a suport a l'aprenentatge autodidàctic. La primera aplicació és un videojoc amb unes escenes bàsiques però riques en diversitat per aprendre a treballar amb l'editor de *Unity* i a realitzar scripts, la segona una aplicació per conèixer i posar en pràctica el paquet *SteamVR* amb el conjunt de dispositius de *HTC Vive*. Aquesta segona aplicació realment té dos objectius, el primer el d'obtenir experiència de cara a la programació amb realitat virtual, i el segon el de servir com a eina de suport als alumnes que comencin el seu treball de final de grau i continuïn el projecte de llarga durada plantejat al *CRV*. Al dissenyar i construir les aplicacions de suport s'han pogut encarar les fases de disseny del *serious game* amb més experiència i així crear-la amb una estructura més compacta i més eines disponibles de treball.

## 8.1 Metodologia de treball

Al dirigir i planejar les tasques que s'havien de fer al projecte primerament es va pensar en una cosa bàsica, s'iniciava amb experiència i coneixements nuls amb el programa *Unity* i això comportava una inversió de temps inicial pel maneig de l'editor i dels scripts que no es podia menysprear, conseqüentment es va veure que era un factor clau distribuir el temps de la feina de la millor manera possible. El millor mètode de treball no podia ser el de començar a crear l'aplicació principal des de zero sense tenir els fonaments bàsics de *Unity* ni de programació amb el llenguatge C#, per això, es va pensar en que el perfecte seria anar implementant els coneixements obtinguts esglaonadament i es va aplicar la idea de separar una primera fase d'aprenentatge i pràctica i una segona fase d'execució dels coneixements obtinguts. Així va ser com es va arrencar amb el desenvolupament del concepte de videojoc *Balls Game*, que després també serviria com a eina d'enllaç a la programació d'entorns de realitat virtuals mitjançant *SteamVR*. A part de l'aprenentatge autodidàctic, també s'havia de tenir en compte el projecte de llarga durada planejat pel *CRV* i així, a més del videojoc es va definir una aplicació que tingués les eines bàsiques de creació d'escenes amb el sistema *HTC Vive*, va sorgir la idea de l'aplicació *HTCViveStudentsCRV.*

Per fer l'anàlisi de l'aportació que han donat cada una de les aplicacions un cop finalitzades al treball, s'ha seguit una estructura de descripció dels apartats on la documentació té un format que separa la informació en tres pilars base de definicions i processos. D'aquesta manera, al segmentar l'anàlisi de les aplicacions amb diferents perspectives i punts de vista, es pot avaluar molt millor quina importància ha tingut cada un dels punts i en el projecte.

Els pilars base de l'anàlisi de l'estructura de cada aplicació de suport són:

- Anàlisi general del videojoc (objectius del joc i explicació de com es juga).
- Estructura interna de funcionament (eines i característiques principals).
- Els coneixement obtinguts després d'haver realitzat l'aplicació (diferenciant entre el mode editor de Unity i les tècniques de programació dels scripts).

## 8.2 L'aplicació d'aprenentatge (*Balls Game*)

Aquesta aplicació va ser la primera que es va dissenyar, crear i programar. L'idea va sorgir d'un tutorial anomenat *Roll a ball* que hi havia a la plataforma de *Unity* per a gent que començava a programar, aquest tutorial consistia en crear una escena amb una pilota i diversos objectes que interactuaven amb ella. L'aplicació *Balls Game* és una continuació de la idea d'aquell tutorial amb diferents escenes, implementacions de noves tècniques amb l'editor de *Unity*, implementació de scripts amb nous tipus de funcions i tot això amb una estructura i format de videojoc. S'han introduït el màxim de diversitat de tècniques d'edició amb *Unity* i programació disponibles per així expandir molt més els coneixements assolits un cop acabada.

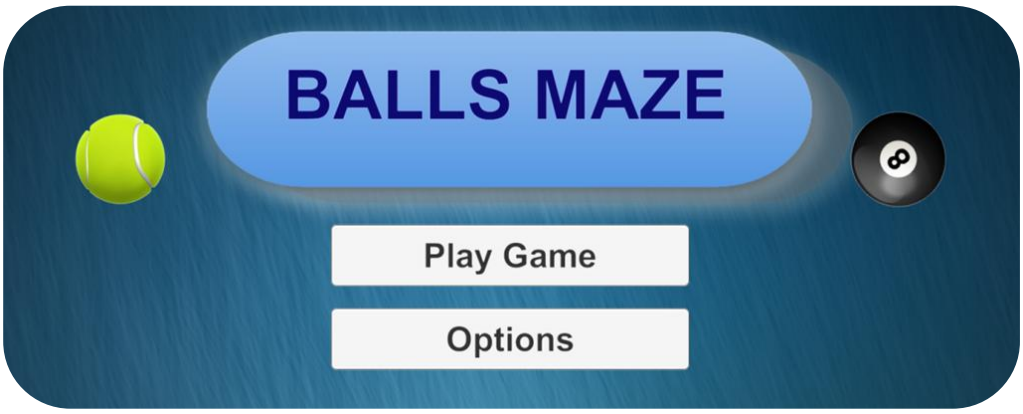



*Figura 8.1: Imatge del menú principal de l'aplicació Balls Game.*

### 8.2.1 Anàlisi del videojoc

*Balls Game* és un videojoc tridimensional en tercera persona que consta de dos minijocs de recol·lecció d'objectes amb temàtica esportiva, el jugador controla pilotes d'un esport en concret i realitza els objectius marcats a cada nivell. Consta d'un menú principal i dos nivells, un nivell basat en el billar i un altre basat en el tennis.

L'estructura de l'aplicació s'ha dissenyat amb una interfície d'usuari com la d'un videojoc convencional, amb la seva pantalla principal, el seu menú d'opcions, el seu menú de tria dels escenaris i la pantalla on expliquen les instruccions de joc de cada nivell, també se li ha introduït música de fons, efecte d'àudio i animacions. Els controls són amb el teclat i el ratolí.

L'estructura base dels dos minijocs és la mateixa, l'objectiu es tracta de moure la pilota per l'escenari per recol·lectar/tocar tots els *pick ups* (que són un tipus d'objectes del videojoc) de cada nivell sense que la pilota toqui els *bad pick ups* (un altre tipus d'objectes del joc). En els dos casos hi ha un límit de temps per realitzar la tasca i hi han diferents tipus de limitacions de l'entorn que, depenent del cas, si es traspassen o bé si es toquen amb la pilota es perd. Abans d'iniciar qualsevol dels dos mini jocs surt la pantalla d'instruccions i una animació on es veu tota l'escena des de l'aire, si es guanya apareix un text conforme s'ha guanyat i si es perd apareix un text amb la causa de la derrota (es pot perdre de vàries maneres diferents).

### 8.2.2 Estructura interna

- Menú principal:

  Dins de l'escena del menú principal es pot navegar entre dues altres pantalles, la de selecció dels nivells i la d'opcions *Figura 8.2* i *Figura 8.3*. A més, els dos nivells del joc tenen un botó que redirigeix al menú principal per si el vol parar de jugar o canviar d'escena. Al menú se li ha afegit àudio (que es pot treure a l'apartat d'opcions) i efectes de so als botons.


*Figura 8.2: Pantalla de selecció de nivells.*


*Figura 8.3: Pantalla d'opcions.*

- Mini joc de billar:

  En el mini joc de billar el jugador controla una bola de billar per una taula. Els *pick ups* són les boles de billar de colors i els *bad pick ups* les boles negres de l'escenari. Si es recol·lecten totes les boles de colors es guanya la partida i si es toca una bola negre o es cau de la taula es perd la partida i el videojoc torna al menú principal.


*Figura 8.4: Imatge del mini joc de billar.*

El moviment de la pilota es realitza amb el teclat (amb les fletxes o amb les tecles *W,A,S,D*), la vista de la càmera és en tercera persona centrada al jugador i varia en funció del moviment del ratolí. Si es mou el ratolí horitzontalment la direcció de la càmera també varia horitzontalment i si es mou el ratolí verticalment la càmera enfoca de més amunt o de més avall.

*Figura 8.5: Límit de camp de visió esquerra.*

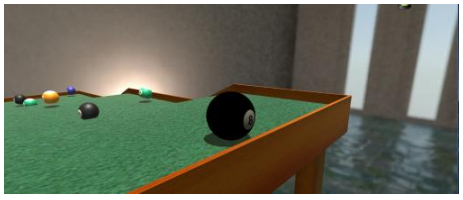


*Figura 8.6: Límit de camp de visió dret.*

*Figura 8.5: Límit de camp de visió inferior.*

*Figura 8.6: Límit de camp de visió superior.*

- Mini joc de tennis:

  En el mini joc de tennis el jugador controla una pilota de tennis en una pista. En aquest cas els *pick ups* són tres dianes on s'ha de rebotar i el *bad pick up* la reixa de la pista de tennis (que es va movent per la pista). La pilota en aquest joc està tota l'estona rebotant amb el terra i quan toques una diana rebota més alt.

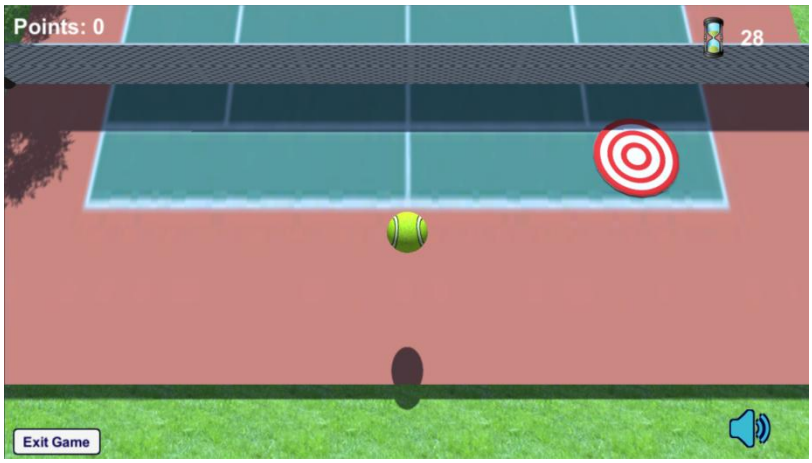



*Figura 8.7: Imatge del mini joc de tennis.*

En aquest cas la càmera no es pot moure amb el ratolí, la càmera, en funció de si s'està a més o menys alçada del terra, s'apropa o s'allunya de la pilota de tennis per crear una percepció d'altura i velocitat. Amb cada diana que es rebota s'arriba a més alçada, al rebotar amb una diana apareix un text *"Boing"* que realitza un moviment de molla (contraient-se i expandint-se) fins a desaparèixer.

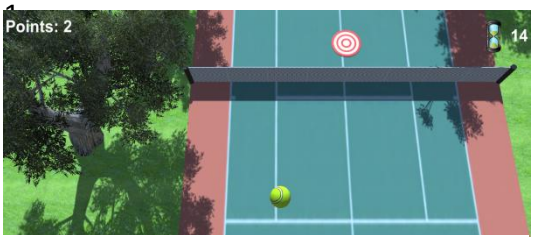



*Figura 8.8: La pilota de tennis a molta altura.*

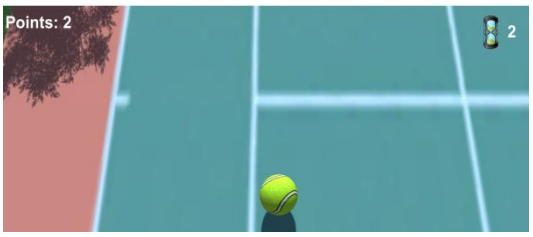



*Figura 8.9: La pilota de tennis a baixa altura.*

A la escena s'han utilitzat modelats *3D* dels arbres amb un *Asset* descarregat de la *Asset Store*. Al igual que el *Skybox* de l'entorn. Els altres objectes estan creats amb l'editor.

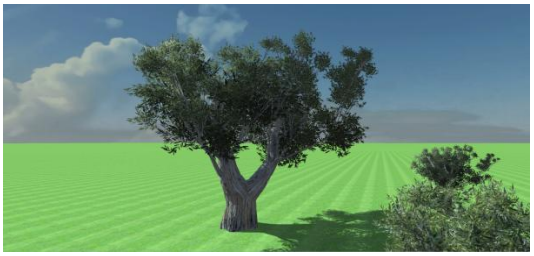


*Figura 8.10: Imatge modelat 3D del arbres.*

ETSEIB

- Interfície d'usuari:

  La interfície d'usuari que es mostra mentre es juga a un dels dos mini jocs és la mateixa en els dos casos. A dalt de la pantalla a l'esquerra es tenen els punts per haver recol·lectat els *pick ups* (cada *pick up* suma 1 punt), a dalt a la dreta el temps límit que falta per finalitzar el joc, a baix a l'esquerra el botó per tornar al menú principal i a baix a la dreta el botó de so per encendre o apagar la música.

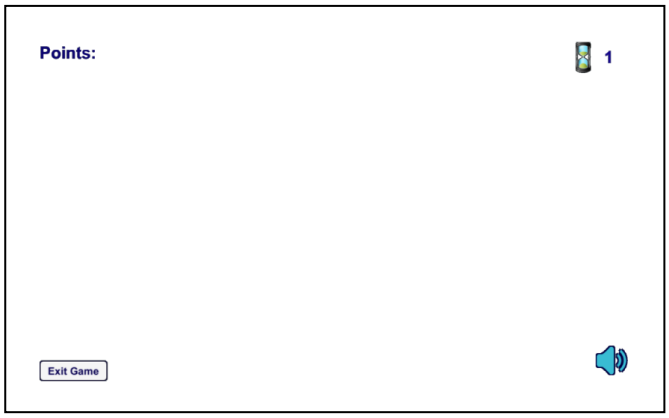



*Figura 8.11: Interfície d'usuari dels dos mini jocs.*

### 8.2.3 Coneixements obtinguts

Els coneixements obtinguts es poden avaluar en dos tipologies diferents, una a nivell d'editor i l'altre a nivell de programació de scripts.

- A nivell de l'editor de *Unity*:

  S'ha aprés a crear animacions bàsiques amb el programa, a crear pistes de so i d'àudio, a crear *skyboxes* i a importar *Prefabs* de la *Asset Store*. S'ha millorat amb el conjunt d'eines de *UI* (botons, texts, imatges, maneig del canvas), en la creació de textures i en el maneig dels *Prefabs*.

- A nivell de scripts:

  S'ha aprés a treballar amb les classes i funcions de so, a treballar amb l'*Scene Manager* (intercanviador d'escenes) i a treballar amb *Colliders*, *Transforms* i forces. En conceptes generals s'ha après a invocar i realitzar totes les funcions bàsiques de *Unity*.

### 8.2.4 Implementació de *HTC Vive* a l'escena

Un cop finalitzada l'aplicació de *Balls Game* es va poder tornar a aprofitar aquesta a mode d'iniciació a l'aprenentatge amb el plug-in *SteamVR* i els dispositius de realitat virtual. Bàsicament la tasca principal que es va plantejar va ser la de poder adaptar *Balls Game* a un entorn de realitat virtual i ser capaç d'utilitzar el *Headset* i els *Controllers* de *HTC Vive* per manejar les escenes. Després també es van realitzar altres accions secundàries per millorar la comoditat dels usuaris a l'hora d'experimentar la immersió virtual del videojoc.

En aquest apartat s'explica com s'han realitzat totes les implementacions des del punt de vista de l'editor de *Unity* i des del punt de vista del scripts.

- Implementacions realitzades amb l'editor:

  Per poder aplicar les escenes a un entorn de realitat virtual es va adaptar des de l'editor el plug-in *SteamVR* de la mateixa manera que s'explica a l'apartat *7.3* del document. Després de realitzar els passos previs de preparació, una de les tasques principals va ser la de canviar els inputs de control perquè en comptes del ratolí i el teclat, es pogués moure al jugador amb el *trackpad* (panell tàctil) dels controladors, més endavant s'explica el script que permet aquesta acció.

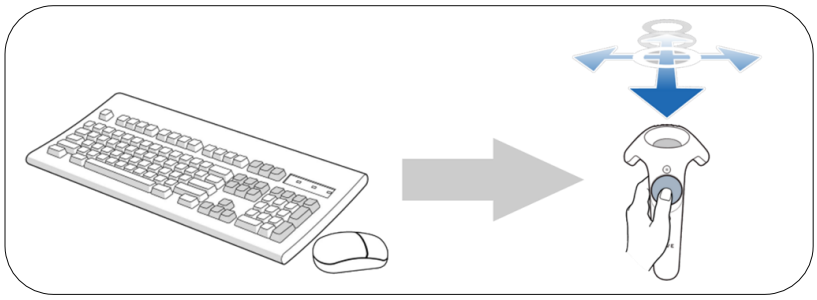


*Figura 8.12: Imatge gràfica del canvi de Inputs realitzats a la escena del projecte.*

Relacionat amb l'editor, també es van fer modificacions respecte la zona de joc de l'usuari: com que ja no es podia tenir una càmera fixada en una zona de l'escenari en concret, degut a que amb el software *SteamVR* la direcció de la càmera depèn totalment del posicionament de les ulleres, es va haver que canviar la posició de la zona d'interacció de les escenes i es van col·locar les *play areas* dels respectius mini jocs en unes ubicacions virtuals on el jugador veiés l'escenari amb comoditat.

*Figura 8.13: Posicionament de l'usuari a l'escena de billar.*

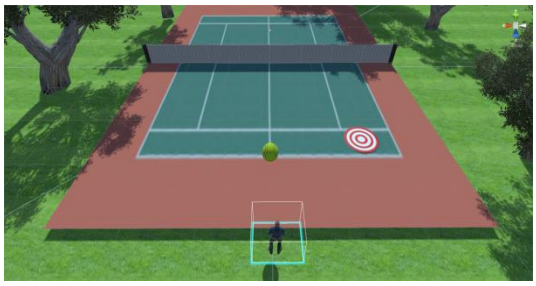


*Figura 8.14: Posicionament de l'usuari a l'escena de tennis.*

A part de la ubicació dels jugadors, a la escena de tennis es va dissenyar un sistema per crear una percepció d'altura i velocitat de la pilota més intensa , elevant l'àrea per on es mou el jugador a la que la pilota passa una certa distància. Imposant aquesta funció es crea una sensació de vertigen que sorprèn als usuaris.

- Implementacions realitzades amb els scripts:

Per enllaçar els inputs dels controladors amb les pilotes, es va programar un script principal anomenat *"Dpad"* que maneja tot el sistema amb el que els controladors interactuen amb el jugador.

El codi intern del programa *Dpad* es va crear amb el suport de dos scripts de *SteamVR* on hi han totes les funcions relacionades amb els *Controllers*. Els scripts que es van utilitzar com a eines de suport són:

- El ***SteamVR_TrackedObject***: Per obtenir la numeració *"índex"* del controlador a l'escena (l'índex és una variable que varia aleatòriament cada vegada que s'inicia l'aplicació i s'utilitza per referenciar els dispositius).
- El ***SteamVR_Controller****:* Per obtenir i invocar els inputs dels controladors. (Els inputs són els botons dels controladors).

S'ha partit l'explicació de funcionament del script *Dpad* en tres fases diferents per poder explicar millor quina funció té cada sector del codi del programa d'una forma més simple i ordenada.

Fase 1: (crida de les variables)

En una primera part del programa s'enllacen totes les variables i els components que es necessiten, aquestes variables són el *GameObject* de la pilota, el seu *Rigidbody* i els scripts de suport de *SteamVR*.

```
public class Dpad : MonoBehaviour
{
SteamVR_TrackedObject controller;
SteamVR_Controller.Device device;

public GameObject ball;
public float force;
private Rigidbody rb;

void Awake()
{
rb = ball.GetComponent<Rigidbody>();
controller = GetComponent<SteamVR_TrackedObject>();
```

Crida dels scripts de *SteamVR*.

Enllaç de l'objecte que es voldrà moure (+la força).

Obtenció dels components interns dels objectes.

*Figura 8.13: Fase1 del script Dpad.*

Les variables *controller* i *device* s'utilitzen com a enllaç a les dades internes dels controladors, la de *controller* serveix per saber l'índex de referència de l'aparell i la *device* per cridar tots els inputs (botons) necessaris.

Fase 2: (crida dels inputs)

En una segona part es crida el paquet d'inputs dels controladors amb la variable *device* i es crea una funció *if* a la que només s'hi entra si es prem el botó *Trigger* (un input dels controladors, a l'apartat *5.2.2* de *Hardware* de *HTC Vive* es poden veure els noms de cada input*)*. A dins de la funció *if* hi ha una línia de codi relacionada amb el *Rigidbody* de la pilota i la seva velocitat que serveix perquè la pilota es pari. En paraules generals, en aquesta part del codi s'ha assignat que quan es premi el botó *Trigger* la pilota es pari.

```
void FixedUpdate()
    {
        device = SteamVR_Controller.Input((int)controller.index);
        if (device.GetTouch(SteamVR_Controller.ButtonMask.Trigger))
        {
            rb.velocity = Vector3.zero;
        }
```

Crida els inputs (botons).

Si es prem el Trigger.

La velocitat del Rigidbody de l'objecte es para.

*Figura 8.14: Fase2 del script Dpad.*

Fase 3: (obtenció de la posició del dit al panell tàctil i aplicació de la força a la pilota) Finalment a la tercera i última fase es realitza una força a la pilota en funció de la zona del *trackpad* que s'estigui tocant.

```
else
        {
            device.GetAxis(Valve.VR.EVRButtonId.k_EButton_SteamVR_Touchpad);

            float x = device.GetAxis(Valve.VR.EVRButtonId.k_EButton_SteamVR_Touchpad).x;
            float z = device.GetAxis(Valve.VR.EVRButtonId.k_EButton_SteamVR_Touchpad).y;

            force = Mathf.Sqrt(x * x + z * z);

            Vector3 direction = new Vector3(x, 0, z);
            rb.AddForce(direction * force);

        }
```

*Figura 8.15: Fase3 del script Dpad.*

La posició on es toca del panell tàctil del *trackpad* esdevé de la funció:

```
device.GetAxis(Valve.VR.EVRButtonId.k_EButton_SteamVR_Touchpad);
```

El valor de la posició en el panell tàctil està emmagatzemada en forma de dos eixos de coordenades *x i y,* on es té un valor màxim de 1 unitat i un valor mínim de -1 unitats, els límits del *Trackpad* es situen es una circumferència de radi 1. A la *Figura 8.16* es mostra la representació gràfica del sistema de coordenades del *Trackpad*, el punt vermell representa el dit de l'usuari i el vector lila el vector que va des de l'origen fins la posició que s'està tocant.

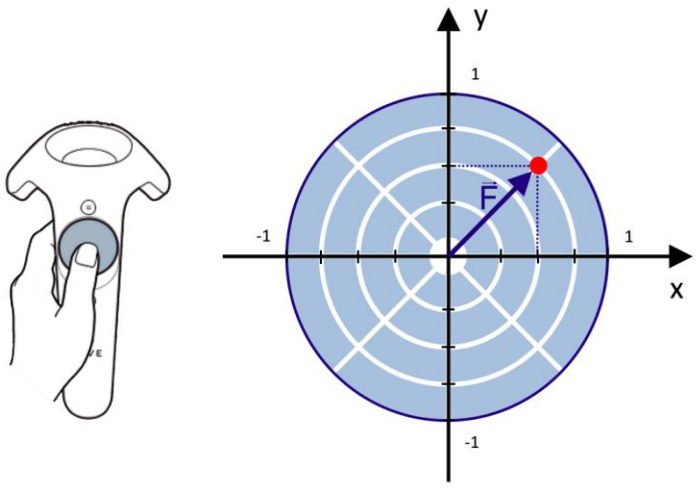



*Figura 8.16: Representació del sistema de coordenades del Trackpad.*

Per aplicar les forces a la pilota el script fa és el següent, primer de tot estableix la direcció del vector per saber cap a on dirigir la pilota, després multiplica aquesta direcció pel mòdul del vector per tenir així una força proporcional a la zona que s'estigui tocant del panell tàctil, tot això ho afegeix com a força al *Rigidbody* de la pilota i aquesta es mou. Si es toca el panell tàctil pels extrems la pilota anirà més ràpid i si es toca pel centre anirà més lenta, així el jugador pot calibrar millor els seus moviments (gràcies a afegir el mòdul del vector dins de la programació).

## 8.3 L'aplicació de suport als alumnes (*HTCViveStudentsCRV*)

### 8.3.1 Anàlisi de l'aplicació

Aquesta aplicació té l'objectiu de proporcionar eines de suport per la interfície de *HTC Vive*. L'aplicació és una sola escena amb el número d'objectes mínims per interactuar i provar les funcions dels scripts desenvolupats.

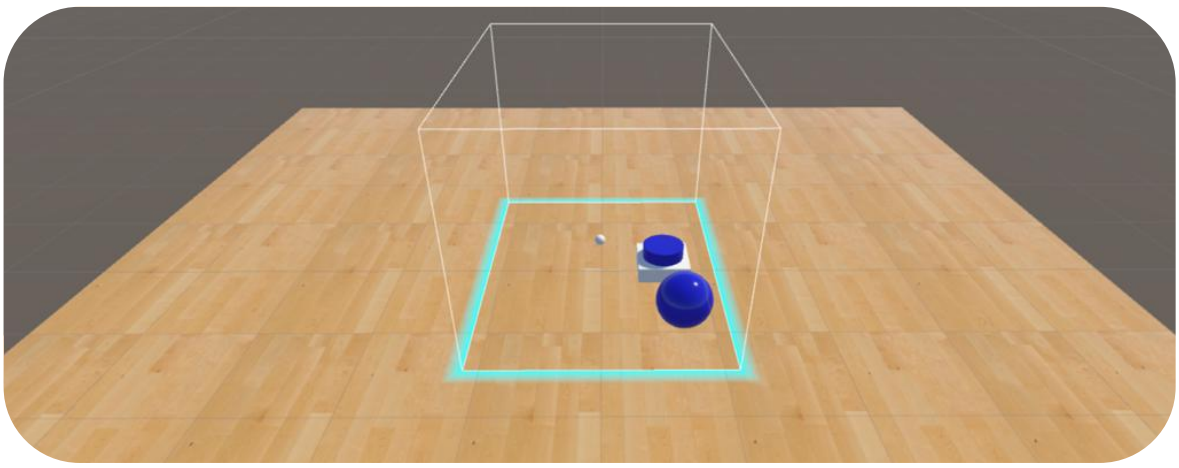


*Figura 8.17: Imatge de la escena HTCViveStudentsCRV*

No s'han volgut posar més coses de les essencials perquè així es simplifica molt més el panell del *Hierarchy* i es poden assimilar més ràpid les funcions que fa cada eina.

### 8.3.2 Estructura interna

L'estructura interna bàsicament són les eines que s'han dissenyat en forma de scripts i *Prefabs*. Les eines s'han guardat en una carpeta anomenada *VRToolsCRV,* en total s'han realitzat sis scripts i dos *Prefabs*, són els següents:

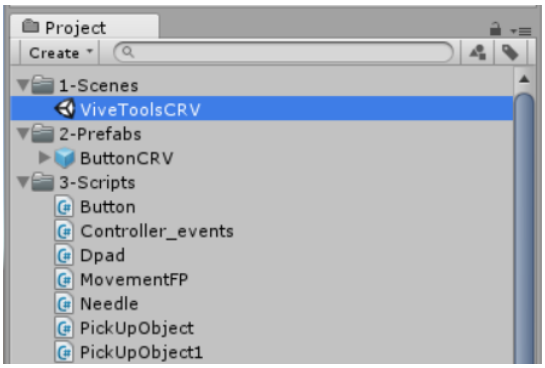



*Figura 8.18: Panell de Project de HTCViveStudentsCRV*

- Scripts:

  - *Dpad:* És el script explicat a l'apartat *8.2.4*, s'utilitza per poder moure objectes amb el *Trackpad* tàctil dels controladors.

  - *MovementFP:* Aquest script s'utilitza per moure's per la pantalla a través del *Trackpad* tàctil, realitza la següent forma de desplaçament: el controlador és la referència de desplaçament (no les ulleres) pel que s'ha de portar el controlador recte en la mateixa direcció cap on es mira. Si el *Trackpad* es tira cap endavant o cap enrere el jugador es mourà en la mateixa direcció (endavant o enrere), i si el *Trackpad* es mou lateralment el jugador girarà cap al costat que toqui. A més, també es pot anar en diagonal, però només en un cert rang fins que es passa a girar sobre el propi eix del jugador com si es toqués lateralment.

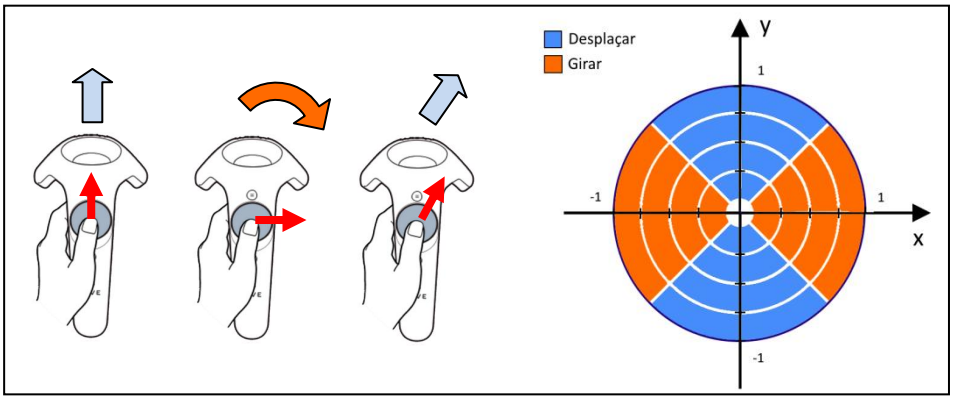



*Figura 8.18: Moviments del jugador en funció del Trackpad.*

- *Needle:* Qualsevol *GameObject* que tingui aquest script i també tingui un *Collider* pot enllaçar un altre objecte perquè només al contactar el destrueixi.
- *PickUpObject:* És un script que s'utilitza per poder agafar objectes amb els controladors. Aquest model agafa i manté agafat un objecte prement el *Trigger* i deixa anar l'objecte al deixar anar el *Trigger*.
- *PickUpObject1:* És un script que s'utilitza per agafar objectes però amb la variació de que s'agafa un objecte i es manté agafat prement una sola vegada el *Trigger* i es deixa l'objecte tornant a prémer el *Trigger*.
- *Controller_*events*:* És un script que ajuda a les persones que vulguin aprendre a utilitzar les diferents funcions de prémer els inputs dels comandaments. Mostra per la consola quan es prem el *Trigger,* quan es manté premut, quan es deixa anar i quan es clica.

A més dels scripts que donen suport a la programació, també hi ha dos prefabs.

- Prefabs:
  - *Prefab de controller (teleport):* És un *Prefab* que serveix per teletransportar-se per l'escenari prement el *Trigger.* Incorpora un làser per fer més fàcil la ubicació de la zona on et teletransporta.
  - *ButtonVR:* Aquest *Prefab* és un botó que es col·loca a l'escenari i activa el que es decideixi. El botó ve amb una animació que s'activa quan es prem i amb un script on se li posen les tasques d'activació.

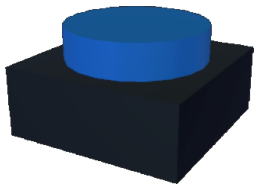


*Figura 8.19:Modelat 3D del Prefab ButtonVR.*

Els botons s'activen a la que interactuen el *Collider* dels controladors amb el *Collider* del botó. No fa falta prémer cap input.

### 8.3.3 Coneixements obtinguts

S'han obtingut tots els coneixements relacionats amb la interacció entre *Unity* i els dispositius de realitat virtual *HTC Vive* amb ajuda de les dades del plug-in *SteamVR*.

# 9. Disseny i construcció de l'aplicació de rehabilitació amb realitat virtual (*MotrizVR*)

Després de la creació de les aplicacions de suport *Balls Game* i *HTCViveStudentsCRV* ja es tenien els coneixements necessaris per endinsar-se en el disseny i construcció de la aplicació final del treball, un *serious game* de rehabilitació de capacitats motores.

## 9.1 Serious game

Un *serious game* és un joc dissenyat amb un propòsit principal diferent al de la pura diversió. Normalment si un joc bé acompanyat de l'adjectiu "seriós" sol ser un producte relacionat amb els sectors de la defensa, l'educació, l'exploració científica, l'enginyeria o, com és el cas d'aquest treball, la sanitat i la rehabilitació de persones.

## 9.2 La realitat virtual com a sistema de rehabilitació

L'ús de la realitat virtual dins del camp de la rehabilitació motora en pacients amb danys cerebrals adquirits *(DCA)* és un fet científicament contrastat en els últims anys. Els prometedors i satisfactoris resultats que s'estan obtenint en aquesta línia d'investigació (tan a nivell cognitiu com a nivell motor), en conjunció amb els processos terapèutics tradicionals, proporcionen l'auge d'una nova vessant de la rehabilitació anomenada *Virtual Motor Rehabilitation (VMR).*

En aquesta última fase del projecte s'ha volgut crear un *serious game* del tipus *VMR* per ajudar a persones amb problemes motors mitjançant la immersió a la realitat virtual. Abans d'iniciar el desenvolupament de la creació de l'aplicació de rehabilitació, es va contactar amb l'enginyer de telecomunicacions i director del centre de la Associació de la Paràlisi Cerebral de Barcelona *ASPACE*, Àngel Aguilar, perquè amb la seva experiència i coneixements donés consells en el procés de disseny dels mini jocs. Es van fer vàries reunions per parlar sobre les maneres més efectives d'implementació, des del punt de vista mèdic, de les ulleres i els controladors de *HTC Vive* a les escenes virtuals i finalment va sortir el disseny final de les estructures dels mini jocs fet en el projecte.

## 9.3 Introducció a l'aplicació

El *serious game* s'ha plantejat i dissenyat com un verdader joc de rehabilitació virtual adreçat a la seva distribució per centres mèdics. Se li ha posat el nom de *MotrizVR* i en ell s'hi integra la màxima cohesió de les tècniques de desenvolupament creades amb el motor de videojocs *Unity* que s'han après durant la fase d'aprenentatge amb les aplicacions de suport.


*Figura 9.1: Logotip de MotrizVR.*

*MotrizVR* ve incorporat amb tres jocs i una zona d'intercanvi entre escenes, les tres escenes de joc serveixen per rehabilitar les capacitats motores de les persones però també són de caire competitiu per fer més entretinguda i amena la pràctica dels exercicis. A més d'etiquetar l'aplicació al gènere de *VMR*, també seria útil com a un videojoc adaptat a nens amb lesions cerebrals permanents, ja que, a diferència dels videojocs comercials de dispositius de moviment, es pot canviar la seva configuració en qualsevol moment per poder-la adaptar a nivells de resposta motrius més lents.

Dues de les aplicacions d'exercicis motrius estan enfocades a la rehabilitació del tronc superior (sobretot braços) i una està enfocada a la rehabilitació cervical (es realitzen les tasques només movent el cap).

## 9.4 Disseny i construcció del joc

Quan s'inicia *MotrizVR,* es comença a l'escena principal d'elecció dels escenaris anomenada *HallScene* i des d'aquesta es pot navegar a qualsevol regió del videojoc. A continuació s'explicarà les funcions i tasques de cada escena individualment. Molts dels tecnicismes no farà falta explicar-los ja que gairebé tots s'han ajustat de tècniques realitzades prèviament al treball. Les escenes de rehabilitació es diuen *Balloons*, *Tunnels i Spaceship.*

### 9.4.1 L'escena principal (*HallScene*)

És la escena que actua com a connector a totes les altres escenes del videojoc, s'ha dissenyat el més simple i senzilla possible perquè no causi confusions a l'hora de l'elecció dels mini jocs. A la *Figura 9.2* es pot veure una imatge del que representa aquest nivell, hi han tres botons per triar alguna de les tres escenes disponibles.

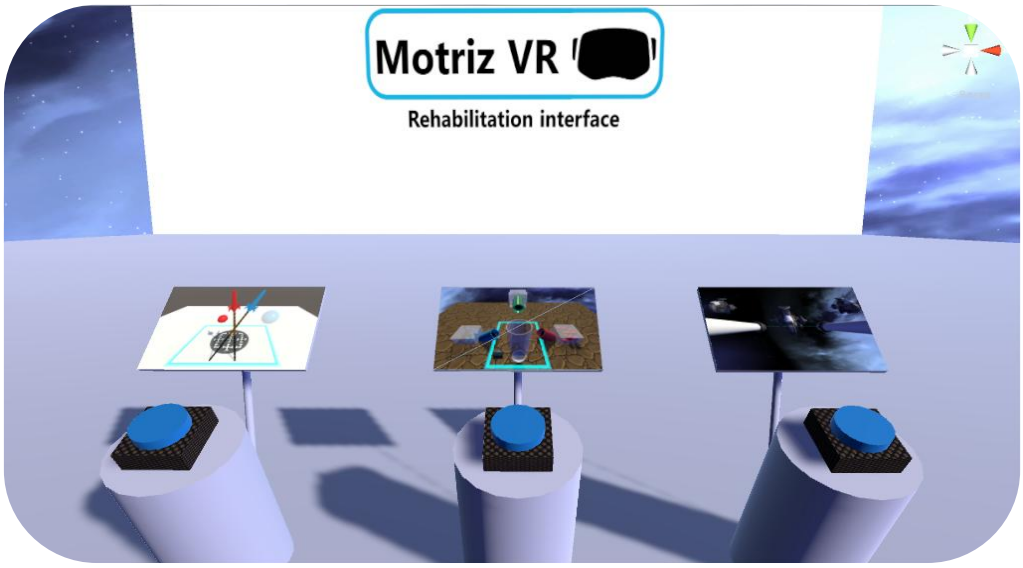



*Figura 9.2: Imatge de la HallScene de MotrizVR.*

S'han inclòs les imatges de cada nivell al davant de cada botó per donar un suport visual i estètic de la pantalla pel jugador. A més, s'ha inclòs el logotip del joc al final de l'escena per donar una mica de *branding* (imatge de marca) del producte.

En una possible actualització de software es posarien les màximes puntuacions obtingudes *Highscores* de cada una de les escenes per fer donar un punt extra de motivació a l'hora de fer els exercicis motors.

### 9.4.2 Escena 1 (*Balloons*)

El mini joc *Balloons* consisteix en explotar dos tipus de globus diferents mitjançant dues llances que es tenen adjuntades a cada *Controller*. Els globus es classifiquen en dos colors, blau i vermell i les llances també, una blava i l'altra vermella.

Només es poder fer explotar els globus si es punxen amb la llança del color corresponent. Al passar un temps predeterminat per un rellotge virtual s'acaba la partida i es torna al menú principal.

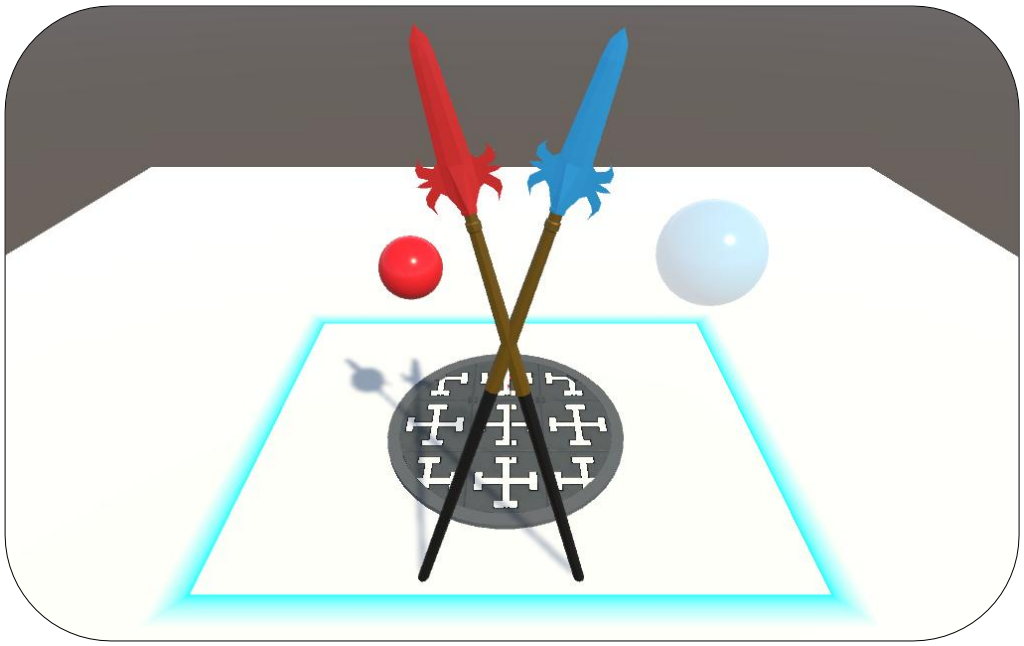


*Figura 9.3: Imatge de l'escena Balloons de MotrizVR.*

Per a jugar correctament al mini joc, el jugador s'ha de posar al cercle central i li aniran apareixen els globus al seu voltant en posicions aleatòries limitades per un volum cilíndric invisible processat amb un script a temps real. S'han tret els render models dels controllers i com a substitució s'hi han posat els modelats de les llances en *3D* per que dona una mica més de sensació d'immersió virtual a dins del joc.

El joc està pensat per fer un exercici de braços amb un rang de moviment molt ampli mentre s'està parat es una mateixa zona, seria un joc molt efectiu com a complement per un pla de rehabilitació per una persona que hagi patit una pèrdua motora relacionada amb el moviment de les espatlles i les seves articulacions.

En una possible actualització de software es posarien les opcions de juga només amb una llança d'un color per si el volés focalitzar la pràctica i millora motora només en un braç.

### 9.4.3 Escena 2 (Tunnels)

L'escena *Tunnels* és en la que es realitzen més moviment de les tres i consisteix en recollir boles de colors d'un recipient per posar-les al túnel del color corresponent.

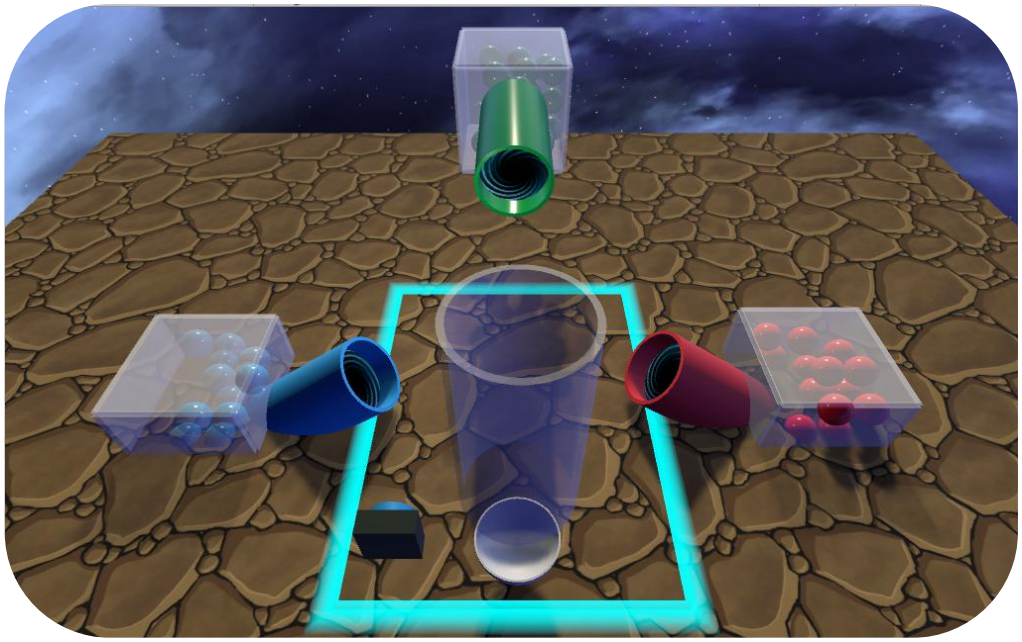


*Figura 9.4: Imatge de l'escenari Tunnels de MotrizVR.*

En aquest escenari es juga només amb una mà, hi ha un controller que està desactivat i d'aquesta manera el pacient no té la temptació d'agafar les boles amb la mà bona. Les boles les fa aparèixer el propi jugador prement un botó que hi ha a l'escena, només surt una bola a la vegada (les boles surten des de dalt del tub transparent fins al bol de sota. *Figura 9.4*
Si es diposita una bola en un tub que no es el correcte apareix un missatge d'error a dins del tub i la bola torna a aparèixer en el recipient de boles, quan es posa una bola al tub corresponent surt un *"+1"* conforme encertes i suma 5 segons al comptador de temps que hi ha a la nivell, si s'acaba el temps es torna a la escena principal.

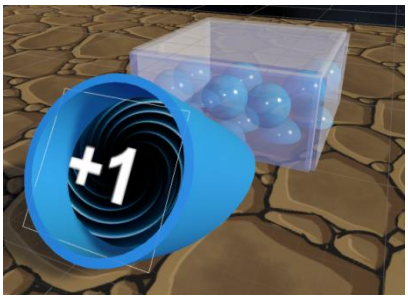



*Figura 9.6: Imatge conforme s'ha posat la bola al tub que toca.*

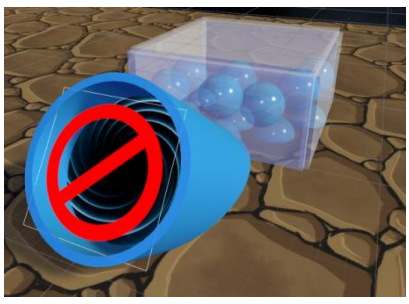


*Figura 9.5: Imatge conforme no s'ha posat la bola al tub que toca.*

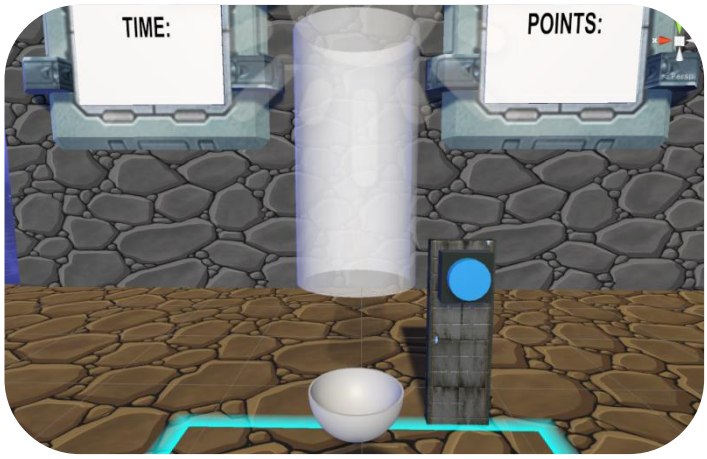



*Figura 9.7: Imatge de l'escena Tunnels del repartidor de boles.*

A la escena es hi han habilitats dos panells de suport al jugador, un amb els punts i l'altre amb el temps restant. Està programat que el mètode d'agafar els objectes sigui amb el *PickObject1* (prémer per agafar i tornar a prémer per deixar) ja que així els pacients han de fer menys forca amb les mans.

És una escena on es realitzen molts moviments d'anada i tornada amb les cames i amb els braços constantment així que seria un joc pensat per persones amb una limitació motora lleu o de corba de recuperació on es veiessin millores ràpides..

En una possible actualització de software s'afegiria un sistema per triar el mètode d'agafar els objectes, podent decidir entre els dos tipus de *PickObjects* i un tercer que fos directament sense prémer cap botó.

### 9.4.4 Escena 3 (Spaceship)

La tercera i última escena consisteix en eliminar naus espacials mitjançant dos canons làsers, és l'única escena en s'interactua només amb el cap. Els canons estan relacionats amb la posició de les ulleres pel que la immersió de realitat virtual que es crea és com si el jugador estigués a dins de la nau espacial disparant. Els canons van disparant contínuament cada x temps rajos làser i l'objectiu consisteix en moure el cap per l'escenari pe apuntar a les naus enemigues que es van movent i eliminar-les. Les naus tenen vida pròpia i cada vegada que se'ls hi resta una de les vides canvien de color (a part tenen el número de vides que els hi queden a dalt seu.

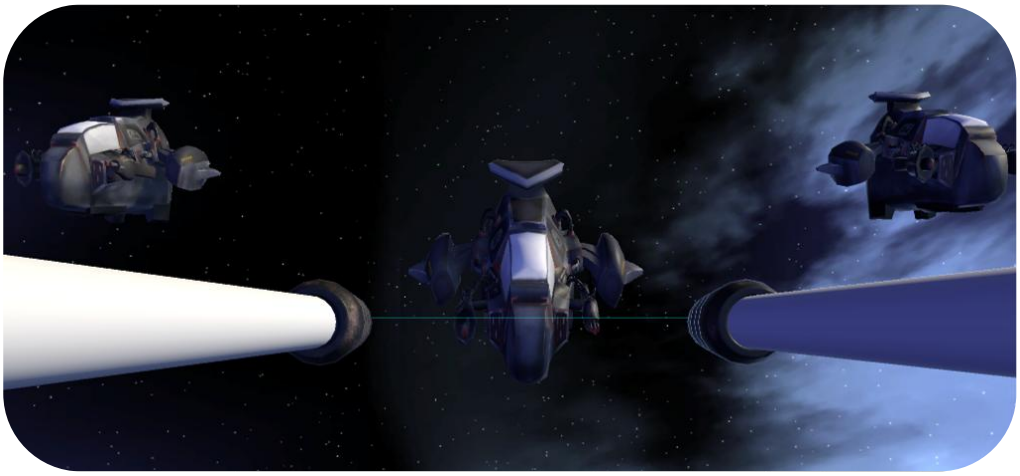


*Figura 9.8: Imatge de l'escena Spaceship de MotrizVR.*

L'escena s'acaba quan es guanya la partida, que és quan el jugador elimina a totes les naus espacials que hi ha a la pantalla.

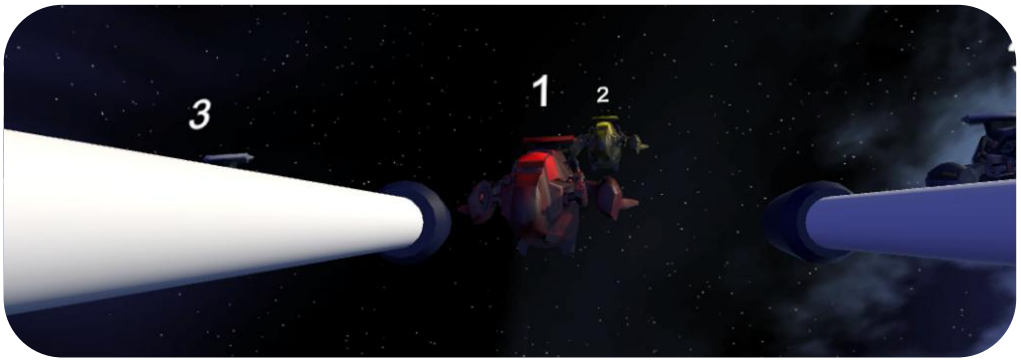



*Figura 9.9: Imatge de l'escena Spaceship en plena partida.*

En una possible actualització de software s'afegiria un sistema per triar el nivell de dificultat del joc i un sistema de punts de vida pel jugador.

# Conclusions

En el que ha durat aquest treball, s'ha passat de no saber res sobre la programació de videojocs i les tecnologies de realitat virtuals a poder acabar realitzant una aplicació completa i relativament complexa sobre el tema. Amb això queda clara una cosa, *Unity* és una sistema de desenvolupament de software que amb pocs coneixements inicials i amb un breu període d'adaptació es poden arribar a crear eines molt útils d'investigació a la realitat virtual.

Per això, la primera conclusió que es pot extreure del treball és que, el sistema ideat pel Centre de Realitat Virtual *CRV* de Barcelona sobre la immersió a la realitat virtual i executat per primera vegada en aquest quadrimestre, si es continua aplicant en els quadrimestres vinents, és més que provable que tingui un futur brillant com a una línia d'investigació als dispositiu d'immersió a la realitat virtual a llarg termini.

La segona conclusió, és que s'ha pogut arribar a crear una aplicació final, que, si s'acabés de perfeccionar en alguns sentits, podria arribar a ajudar a la verdadera rehabilitació de persones que ho necessitessin i això és una cosa molt positiva que s'ha de valorar. No fa res des de que la finestra del món a la realitat virtual ha aparegut com a una eina a tenir en compte a les nostres vides, de moment és un sistema molt nou, però veient el potencial que pot arribar a tenir, s'ha de continuar avançant perquè això no quedi aquí i s'expandeixi, evolucioni i millori la tecnologia de mica en mica a més entorns de la nostre societat.

# Agraïments

Primer de tot, agrair al tutor del treball, Toni Susin, per la magnífica oportunitat que ens ha brindat als quatre alumnes del quadrimestre, de poder obrir el camí a seguir d'aquesta nova i interessant línia de projectes de realitat virtual.

Seguidament agrair a Jordi Moyés, qui ha estat sempre a disposició per qualsevol tema de la disponibilitat de les sales on es treballava amb els dispositius del Centre de Realitat Virtual (especialment agraït per l'esforç que van fer ell i en Toni les últimes setmanes del projecte).

I finalment donar també un agraïment especial a la Dra. Anna Fornós, neuròloga i pediatra, directora assistencial de *ASPACE (centre de la Associació de Paràlisi Cerebral de Barcelona)* i a Àngel Aguilar, enginyer de telecomunicacions i director de *ASPACE,* per tot el suport que m'han donat amb el tema de la rehabilitació motora del projecte a mesura que es dissenyaven les aplicacions.

# Bibliografia

## Referències bibliogràfiques

**[1]** MAMMUT, Tratatto della Pittura, Leonardo Da Vinci

**[2]** Sutherland, Ivan: (1965). *The Ultimate Displa*y. IFIP Congress.

## Bibliografia complementària

- OXFORD PSYCHOLOGY SERIES NO.20, Binocular Vision and Stereopsis, Ian P. Howard, Brian J. Rogers
- HISTORY OF TECHNOLOGY SERIES NO.29, Sir Charles Wheatstone FRS 1802-1875, Brian Bowes
- WILEY-INTERSCIENCE, Virtual Reality Technology, Grigore C. Burdea, Philippe Coiffet
- UNIVERSITY OF CAMBRIDGE, Sketchpad: A man-machine graphical communication system, Ivan Edward Sutherland
- Proceedings of IFIP Congress, The Ultimate Display, Ivan Edward Sutherland
- ACADEMIC PRESS, Virtual Reality systems, R.A. Earnshaw, M.A. Gigante, H.Jones
- O'RELLY, Learning Virtual Reality, Tony Parisi